ComponentOne

DateTimeEditors for WPF and Silverlight

GrapeCity US

GrapeCity
201 South Highland Avenue, Suite 301
Pittsburgh, PA 15206
Tel: 1.800.858.2739 | 412.681.4343
Fax: 412.681.4384
Website: https://www.grapecity.com/en/
E-mail: us.sales@grapecity.com

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is 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 media 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 the defective media by sending it and a check for $25 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is 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. ComponentOne is 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

Table of Contents

DateTimeEditors for WPF and Silverlight Overview

Display, edit and validate DateTime information using **DateTime Editors for WPF and Silverlight**. The C1DateTimePicker control provides a single, intuitive UI for selecting date and time values. The C1TimeEditor control provides a simple masked editor for just time values. Users can edit date and time values using the spin buttons, keyboard arrows, or by typing in fields.

Help with WPF and Silverlight Edition

Getting Started

- For information on installing **ComponentOne Studio WPF Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with WPF Edition.
- For information on installing **ComponentOne Studio Silverlight Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with Silverlight Edition.

C1DateTimeEditors Key Features

DateTimeEditors for WPF and Silverlight allows you to create customized, rich applications. Make the most of **DateTimeEditors for WPF and Silverlight** by taking advantage of the following key features:

- **Multiple Display Versions**
  Choose one of the available edit modes: **DateTime (default)**, **Date**, or **Time**. Select from preset date formats, including **Short** and **Long**. Choose from preset time formats, including **ShortTime**, **LongTime**, and **TimeSpan**.
- **Supports Spin Buttons**
  The C1DateTimePicker and C1TimeEditor controls support spin (up/down) buttons for selecting date and time.
- **Wide Range of Cultures**
  Define the cultural setting for date time formats.
- **Supports Null Values**
  The **C1DateTimePicker**, C1DatePicker and **C1TimeEditor** controls allow you to enter null values, by default. This can be disabled by setting the AllowNull property to **False**.

XAML Quick Reference

This topic is dedicated to providing a quick overview of the XAML used to create a C1DateTimePicker control. The XAML markup in this section illustrates how to create a **C1DateTimePicker** control with its date and time set and its date format set.

```
<c1datetime:C1DateTimePicker DateTime="1/17/2010 11:04 AM" DateFormat="Long">
```

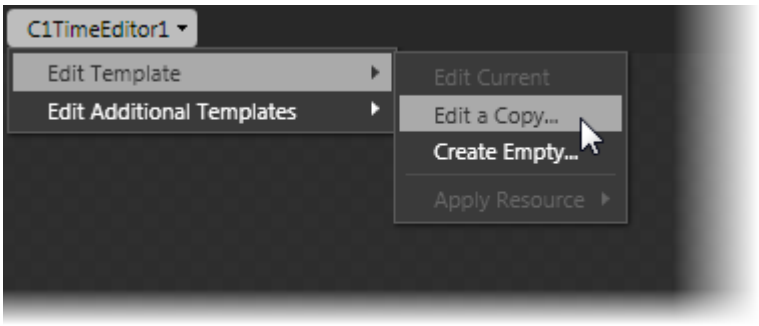
Templates (Silverlight)

One of the main advantages to using a Silverlight control is that controls are "lookless" with a fully customizable user interface. Just as you design your own user interface (UI), or look and feel, for Silverlight applications, you can provide your own UI for data managed by **DateTimePicker for Silverlight**. Extensible Application Markup Language (XAML;

pronounced "Zammel"), an XML-based declarative language, offers a simple approach to designing your UI without having to write code.

Accessing Templates

You can access templates in Microsoft Expression Blend by selecting the C1DateTimePicker, C1DatePicker, or C1TimeEditor control and, in the menu, selecting **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or **Create Empty** to create a new blank template.



Note: If you create a new template through the menu, the template will automatically be linked to that template's property. If you manually create a template in XAML you will have to link the appropriate template property to the template you've created.

Note: You can use the **Template** property to customize the template.

C1DateTimePicker Control Help

Exchange date and time information using **DateTimePicker for WPF and Silverlight**. It provides a simple and intuitive UI for selecting date and time or just time values. The date and time can be selected by using the spin buttons, keyboard arrows, or by typing in fields.

C1DateTimePicker Quick Start

The following quick start guide is intended to get you up and running with C1DateTimePicker. In this quick start, you'll start in Visual Studio to create a new project, add a **C1DateTimePicker** control to your application, and customize the **C1DateTimePicker** control.

Step 1 of 3: Creating an Application with a C1DateTimePicker Control

In this step, you'll begin in Visual Studio to create a WPF or Silverlight application using C1DateTimePicker.

Complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select **WPF Application** or **Silverlight Application**.
3. Enter a **Name** and **Location** for your project and click **OK** to create the new application.
4. If you created a WPF project, in the Toolbox, double-click the **C1DateTimePicker** icon to add the **C1DateTimePicker** control to the WPF application.
   If you created a Silverlight project, follow these steps:
   a. In the XAML window of the project, resize the **UserControl** by changing **DesignWidth="400" DesignHeight="300"** to **DesignWidth="Auto" DesignHeight="Auto"** in the tag so that it appears similar to the following:

   XAML

   ```
   <UserControl x:Class="SilverlightApplication24.MainPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:d=http://schemas.microsoft.com/expression/blend/2008
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
   mc:Ignorable="d" d:DesignWidth="Auto" d:DesignHeight="Auto">
   ```

   The **UserControl** will now resize to accommodate any content placed within it.

   b. In the XAML window of the project, place the cursor between the <*Grid*> and <*/Grid*> tags and click once. Note that you cannot currently add Silverlight controls directly to the design area in Visual Studio, so you must add them to the XAML window as directed in the next step.
   c. Navigate to the Toolbox and double-click the C1DateTimePicker icon to add the control to the grid. The XAML markup resembles the following:

   XAML

   ```
   <Grid x:Name="LayoutRoot">
       <c1datetime:C1DateTimePicker></c1datetime:C1DateTimePicker>
   ```

```
</Grid>
```

You have completed the first step of the **C1DateTimePicker** quick start. In this step, you created a project and added a **C1DateTimePicker** control to it. In the next step, you'll customize the control.

Step 2 of 3: Customizing the C1DateTimePicker

In the previous step, you created a WPF or Silverlight application with a C1DateTimePicker control. In this step, you will modify the appearance of the control.

WPF

Select the **C1DateTimePicker** control and then, in the Properties window, set the following properties:

- Set the **Height** property to "30" to set height of the control.
- Set the **Width** property to "300" to set the width of the control.
- Set the TimeFormat property to **ShortTime** to change the format of the time to a short format consisting of only hours and minute spaces.
- Set the DateFormat property to **Long** to change the format of the date to a longer format that includes the weekday.
- Set the SelectionBackground property to **LimeGreen** to modify the color of the control's selected area.
- Set the FirstDayOfWeek property to **Wednesday** to change the first day of the drop-down calendar's week to Wednesday.

Silverlight

Complete the following steps:

1. Add **Height="30"** to the *<c1datetime:C1DateTimePicker>* tag to determine height of the control. The XAML markup appears as follows:
```
<c1datetime:C1DateTimePicker Height="30">
```
2. Add **Width="300"** to the *<c1datetime:C1DateTimePicker>* tag to determine the width of the control. The XAML markup appears as follows:
```
<c1datetime:C1DateTimePicker Height="30" Width="300">
```
3. Add **TimeFormat="ShortTime"** to the *<c1datetime:C1DateTimePicker>* tag to change the format of the time to a short format consisting of only hours and minute spaces. The XAML markup appears as follows:
```
<c1datetime:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime">
```
4. Add **DateFormat="Long"** to the *<c1datetime:C1DateTimePicker>* tag to change the format of the date to a longer format that includes the weekday. The XAML markup appears as follows:
```
<c1datetime:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime"
 DateFormat="Long">
```
5. Add **SelectionBackground="LimeGreen"** to the *<c1datetime:C1DateTimePicker>* tag; this will modify the color of the selected area of the C1DateTimePicker control. The XAML markup appears as follows:
```
<c1datetime:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime"
 DateFormat="Long" SelectionBackground="LimeGreen">
```
6. Add **FirstDayOfWeek="Wednesday"** to the *<c1datetime:C1DateTimePicker>* tag; this will change the first day of the drop-down calendar's week to Wednesday. The XAML markup appears as follows:
```
<c1datetime:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime"
 DateFormat="Long" SelectionBackground="LimeGreen" FirstDayOfWeek="Wednesday">
```

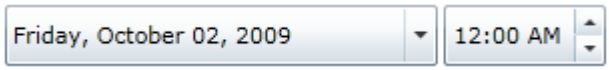
In this step, you customized the appearance of the **C1DateTimePicker** control. In the next step, you will run the project and experience the functionality of the control.

Step 3 of 3: Running the Project

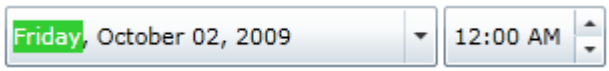
In the last step, you customized the C1DateTimePicker control. In this step, you will run the project and observe some of the run-time features of the control.

Complete the following steps:

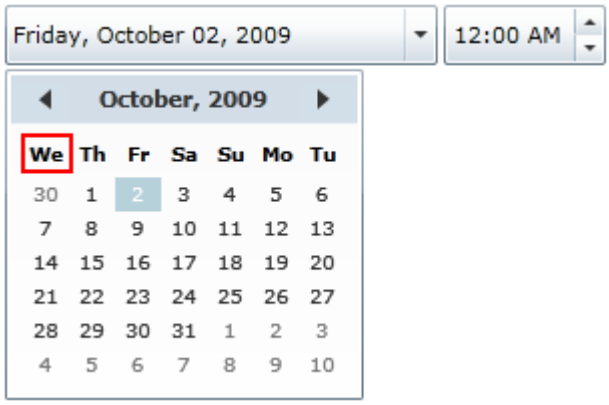
1. From the toolbar menu, select **Project | Run Project** to run your application. Observe that the video content plays automatically and that the application resembles the following:



2. Using your cursor, highlight an area in the date picker. The selection resembles the following:



3. Click the time picker drop-down arrow to reveal the calendar and observe that the calendar's weeks start with Wednesday:



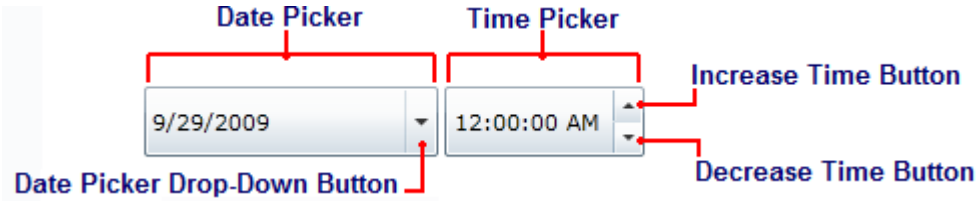
Congratulations! You have completed the **C1DateTimePicker** quick start. In this quick start, you created a WPF application containing a **C1DateTimePicker** control, modified the control's appearance, Now that you have finished the quick start, we recommend that you visit the Working with C1DateTimePicker or C1DateTimePicker Task-Based Help topics.

Working with C1DateTimePicker

The following topics will provide you with an overview of the C1DateTimePicker control's elements and features.

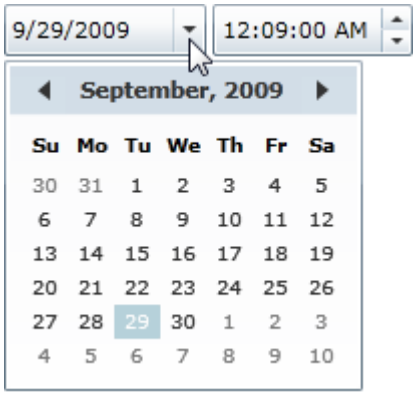
C1DateTimePicker Elements

DateTimeEditors for WPF and Silverlight includes the C1DateTimePicker control, a simple control which provides, by default, both a date picker and a time picker. When you add the **C1DateTimePicker** control to a XAML window, it exists as a completely functional date and time picker. By default, the control's interface looks similar to the following image:



The **C1DateTimePicker** control consists of the following elements:

- **Date Picker**
  The date picker element is comprised of a date field and the calendar drop-down button. You can set the date by entering numeric values or by selecting a date from the calendar.
- **Time Picker**
  The time picker element is comprised of the time field, the increase time button, and the decrease time button. You can set the time by entering numeric values or by clicking the buttons.
- **Date Picker Drop-Down Button**
  The date picker drop-down button opens a calendar from where you can select a date for the date picker.



- **Increase Time Button**

  The increase time button allows you to increase the time displayed in the time picker. Clicking the increase button will increase the time by one minute.

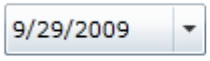
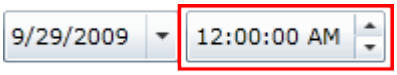
- **Decrease Time Button**

  The decrease time button allows you to decrease the time displayed in the time picker. Clicking the decrease button will decrease the time by one minute.

C1DateTimePicker Edit Modes

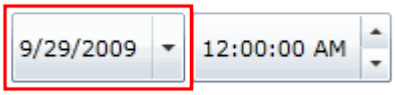
By default, the C1DateTimePicker control will appear on the page with both the date picker and the time picker. You can change the pickers that are displayed by setting the C1DateTimePicker.EditMode property to **Date**, **Time**, or

DateTime. You can set the **EditMode** property to **Date** to display only the date picker; you can set the **EditMode** property to **Time** to only display the time picker; and you can set the **EditMode** property to **DateTime** to display both the time picker and date picker. The table below illustrates each editor mode.

Editor Mode	Result
Date	
Time	
DateTime	

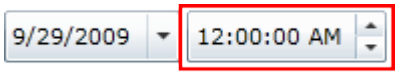
C1DateTimePicker Date Format

You can use the C1DateTimePicker.DateFormat property to set the format that the date picker displays. You can set **DateFormat** property to either **Short** or **Long**. The table below illustrates the two date formats.

Date Format	Result
Short (Default)	
Long	

C1DateTimePicker Time Format

You can use the C1DateTimePicker.TimeFormat property to set the format that the date picker displays. You can set **TimeFormat** property to either **ShortTime** or **LongTime**. The table below illustrates the two date formats.

Time Format	Result
ShortTime	
LongTime (default)	

C1DateTimePicker for WPF Layout and Appearance

DateTimeEditors for WPF and Silverlight 11

The following topics detail how to customize the C1DateTimePicker control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases. Themes allow you to customize the appearance of the grid and take advantage of Silverlight's XAML-based styling. You can also use templates to format and layout the control and to customize the control's actions.

C1DateTimePicker ClearStyle Properties

DateTimePicker for WPF and Silverlight supports ComponentOne's new ClearStyle technology that allows you to easily change control colors without having to change control templates. By just setting a few color properties you can quickly style the entire grid.

The following table outlines the brush properties of the C1DateTimePicker control:

Brush	Description
Background	Gets or sets the brush of the control's background.
ButtonBackground	Gets or sets the brush of the buttons' background colors.
ButtonForeground	Gets or sets the brush of the buttons' foreground colors.
MouseOverBrush	Gets or sets the System.Windows.Media.Brush used to highlight the buttons when the mouse is hovered over them.
PressedBrush	Gets or sets the System.Windows.Media.Brush used to highlight the buttons when they are clicked on.

You can completely change the appearance of the **C1DateTimePicker** control by setting a few properties, such as the ButtonBackground property, which sets the background of the drop-down arrow, for the **C1DateTimePicker** control. For example, if you set the **ButtonBackground** property to "#FFC500FF", the **C1DateTimePicker** control would appear similar to the following:

C1DateTimePicker Appearance Properties

The **C1DateTimePicker** control includes several properties that allow you to customize the appearance of the control. You can change the appearance of the text displayed in the control and customize graphic elements of the control. The following tables describe some of these appearance properties.

Text Properties

The following properties allow you to customize the appearance of text in the C1DateTimePicker control.

Property	Description
FontFamily	Gets or sets the font family of the control. This is a dependency property.
FontSize	Gets or sets the font size. This is a dependency property.
FontStretch	Gets or sets the degree to which a font is condensed or expanded on the screen. This is a dependency property.
FontStyle	Gets or sets the font style. This is a dependency property.
FontWeight	Gets or sets the weight or thickness of the specified font. This is a dependency property.

Color Properties

The following properties allow you to customize the colors used in the control itself.

Property	Description
Background	Gets or sets a brush that describes the background of a control. This is a dependency property.
Foreground	Gets or sets a brush that describes the foreground color. This is a dependency property.

Border Properties

The following properties let you customize the control's border.

Property	Description
BorderBrush	Gets or sets a brush that describes the border background of a control. This is a dependency property.
BorderThickness	Gets or sets the border thickness of a control. This is a dependency property.

Size Properties

The following properties let you customize the size of the C1DateTimePicker control.

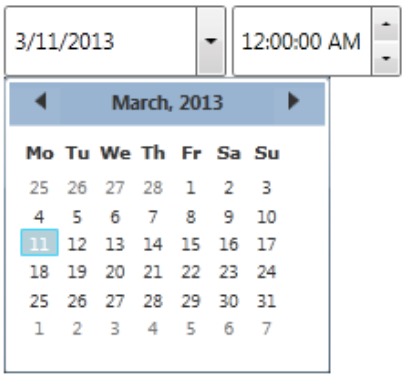
Property	Description

ActualHeight	Gets the rendered height of this element. This is a dependency property.
ActualWidth	Gets the rendered width of this element. This is a dependency property.
Height	Gets or sets the suggested height of the element. This is a dependency property.
MaxHeight	Gets or sets the maximum height constraint of the element. This is a dependency property.
MaxWidth	Gets or sets the maximum width constraint of the element. This is a dependency property.
MinHeight	Gets or sets the minimum height constraint of the element. This is a dependency property.
MinWidth	Gets or sets the minimum width constraint of the element. This is a dependency property.
Width	Gets or sets the width of the element. This is a dependency property.

C1DateTimePicker Theming

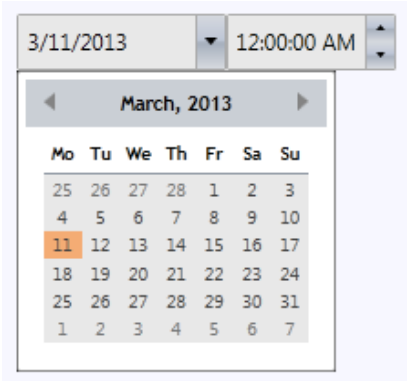
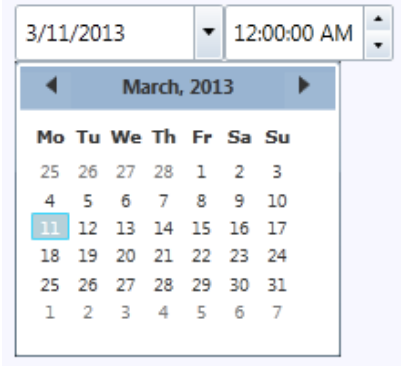
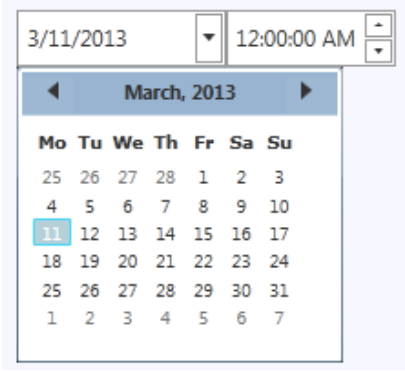
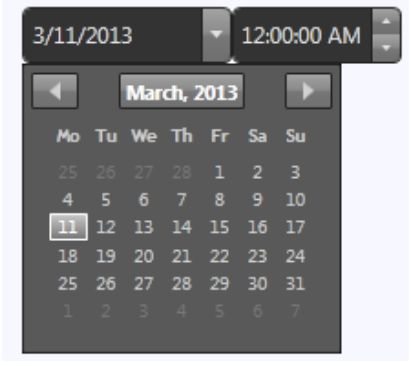
Themes are a collection of image settings that define the look of a control or controls. The benefit of using themes is that you can apply the theme across several controls in the application, thus providing consistency without having to repeat styling tasks.

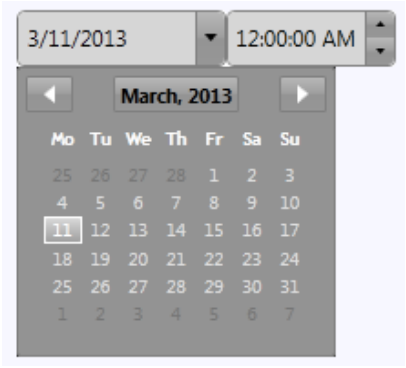
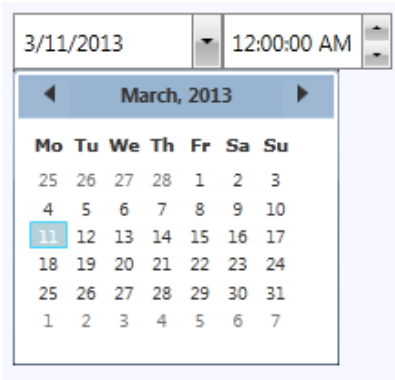
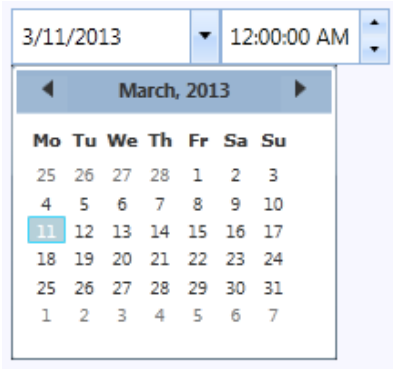
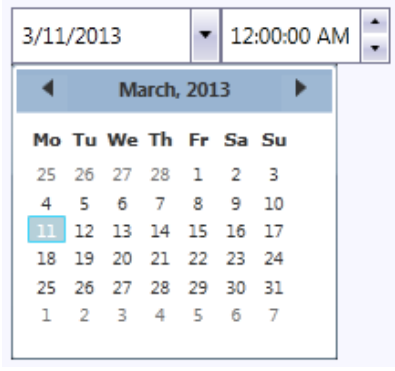
When the C1DateTimePicker control is added to your project, it appears with the default theme, which looks as follows:

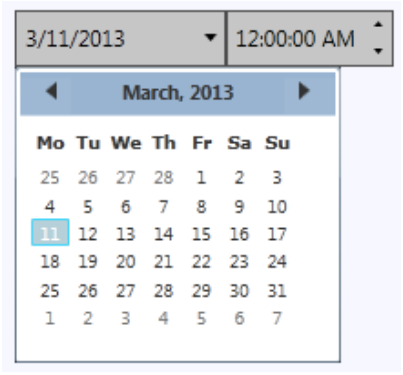
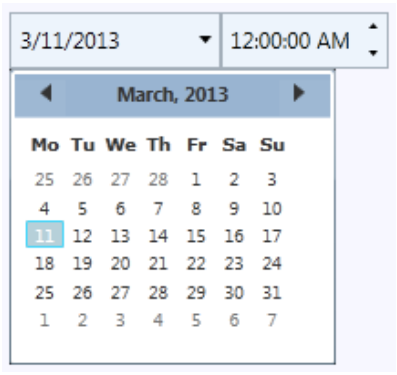
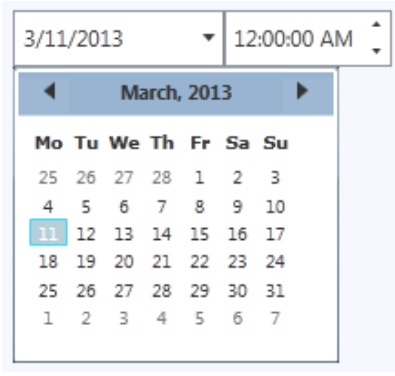
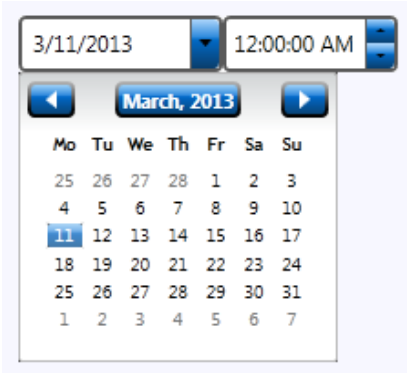


But the **C1DateTimePicker** control can also be themed with one of our thirteen included WPF themes: *BureauBlack, C1Blue, ExpressionDark, ExpressionLight, Office2007Black, Office2007Blue, Office2007Silver, Office2010Black, Office2010Blue, Office2010Silver, ShinyBlue,* and *WhistlerBlue*. The table below provides a sample of each theme:

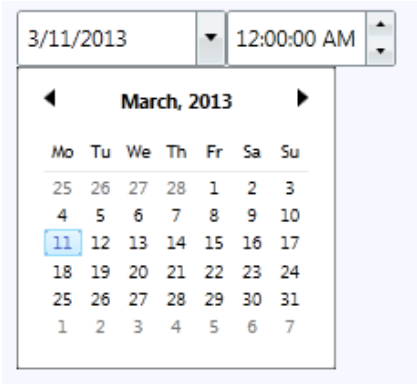
Theme Name	Appearance

BureauBlack	
C1Blue	
Cosmopolitan	
ExpressionDark	

ExpressionLight	
Office2007Black	
Office2007Blue	
Office2007Silver	

Office2010Black	
Office2010Blue	
Office2010Silver	
ShinyBlue	

WhistlerBlue



To set an element's theme, use the **ApplyTheme** method. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
   Dim theme As New C1ThemeExpressionDark
   ' Using ApplyTheme
   C1Theme.ApplyTheme(LayoutRoot, theme)
```

C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
   {
      C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
      //Using ApplyTheme
      C1Theme.ApplyTheme(LayoutRoot, theme);
   }
```

To apply a theme to the entire application, use the **System.Windows.ResourceDictionary.MergedDictionaries** property. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
   Dim theme As New C1ThemeExpressionDark
   ' Using Merged Dictionaries
Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme))
End Sub
```

C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
   {
      C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
      //Using Merged Dictionaries
Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme));
   }
```

Note that this method works only when you apply a theme for the first time. If you want to switch to another ComponentOne theme, first remove the previous theme from **Application.Current.Resources.MergedDictionaries**.

C1DateTimePicker Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio and know how to use the C1DateTimePicker control in general. If you are unfamiliar with the **C1DateTimePicker** control, please see the C1DateTimePicker Quick Start first.

Each topic in this section provides a solution for specific tasks using the **C1DateTimePicker** product.

Each task-based help topic also assumes that you have created a new WPF project.

Allowing Null Values

By default, the C1DateTimePicker control doesn't allow users to enter null values, but you can force the control to accept a null value by setting the C1DateTimePicker.AllowNull property to **True**. In this topic, you will learn how to set the **AllowNull** property to **True** in the designer, in XAML, and in code.

In the Designer

Complete the following steps:

1. Click the **C1DateTimePicker** control once to select it.
2. In the **Properties** window, select the **C1DateTimePicker.AllowNull** check box.

In XAML

To allow null values, place **AllowNull="True"** to the *<my:C1DateTimePicker>* tag so that the markup resembles the following:

```
<my:C1DateTimePicker AllowNull="True"/>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1DateTimePicker1.AllowNull = True
```

C#

```
c1DateTimePicker1.AllowNull = true;
```

3. Run the project.

Tip: to set null value at run-time, users should either clear all the text in the **DatePicker** portion of the control and press ENTER or move focus to the **TimeEditor** portion of the control.

Selecting the Edit Mode

By default, the C1DateTimePicker control shows both the date and time pickers, but you may also choose to show only the date picker or only the time picker. In this topic, you will learn how to change the editor mode in the designer, in XAML, and in code.

In the Designer

To change the edit mode, complete the following steps:

1. Click the **C1DateTimePicker** control once to select it.
2. In the **Properties** window, click the C1DateTimePicker.EditMode drop-down arrow and select a mode from the list. For this example, select **Date**.

In XAML

To change the edit mode, place **EditMode="Date"** to the *<my:C1DateTimePicker>* tag so that the markup resembles the following:

```
<my:C1DateTimePicker EditMode="Date">
```

In Code

To change the edit mode, complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Import the following namespace:

Visual Basic

```
Imports C1.WPF.DateTimeEditors
```

C#

```
using C1.WPF.DateTimeEditors;
```

3. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1DateTimePicker1.EditMode = C1DateTimePickerEditMode.Date
```

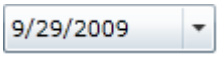
C#

```
c1DateTimePicker1.EditMode = C1DateTimePickerEditMode.Date;
```

4. Run the project.

This Topic Illustrates the Following:

In this topic, you set the C1DateTimePicker.EditMode to **Date**, which removes the time picker form the **C1DateTimePicker** control. The result of this topic will resemble the following image:



Selecting the Time Format

By default, the C1DateTimePicker control's time picker displays the time in a long format that includes seconds, but it can also display time in a shorter format. In this topic, you will learn how to change the time format in the designer, in XAML, and in code.

In the Designer

To change the time format, complete the following steps:

1. Click the **C1DateTimePicker** control once to select it.
2. In the **Properties** window, click the C1DateTimePicker.TimeFormat drop-down arrow and select a mode from the list. For this example, select **ShortTime**.

In XAML

To change the time format, place **TimeFormat="ShortTime"** to the *<my:C1DateTimePicker>* tag so that the markup resembles the following:

```
<my:C1DateTimePicker TimeFormat="ShortTime">
```

In Code

To change the time format, complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Import the following namespace:

Visual Basic

```
Imports C1.WPF.DateTimeEditors
```

C#

```
using C1.WPF.DateTimeEditors;
```

3. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1DateTimePicker1.TimeFormat = C1TimeEditorFormat.ShortTime
```

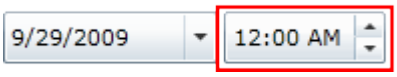
C#

```
c1DateTimePicker1.TimeFormat = C1TimeEditorFormat.ShortTime;
```

4. Run the project.

This Topic Illustrates the Following:

In this topic, you set the **TimeFormat** to **ShortTime**, which provides a shortened time display. The final result will resemble the following image:



Selecting the Date Format

By default, the C1DateTimePicker control's date picker displays the date in a short format, but it can also display time in a long format. In this topic, you will learn how to change the date format in the designer, in XAML, and in code.

In the Designer

To change the date format, complete the following steps:

1. Click the C1DateTimePicker control once to select it.
2. In the **Properties** window, click the C1DateTimePicker.DateFormat drop-down arrow and select a mode from the list. For this example, select **Long**.

In XAML

To change the date format, place **DateFormat="Long"** to the *<my:C1DateTimePicker>* tag so that the markup resembles the following:

```
<my:C1DateTimePicker DateFormat="Long">
```

In Code

To change the date format, complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1DateTimePicker1.DateFormat = Microsoft.Windows.Controls.DatePickerFormat.Long
```

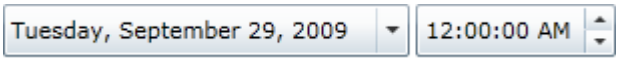
C#

```
c1DateTimePicker1.DateFormat = Microsoft.Windows.Controls.DatePickerFormat.Long;
```

3. Run the project.

This Topic Illustrates the Following:

In this topic, you set the **DateFormat** to **Long**, which will display the date in a long format. The final result will resemble the following image:



Setting the Minimum and Maximum Calendar Dates

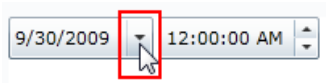
You can change the dates that calendar spans by setting the **MinimumDate** and **MaximumDate** properties in the designer, in XAML, and in code.

> **Note**: Try to avoid avoid setting the C1DateTimePicker.MinDate and C1DateTimePicker.MaxDate properties in XAML as a string value. Parsing these values from strings is culture specific. If you set a value with your current culture and a user is using different culture, the user can get **XamlParseException** when loading your site. The best practice is to set these values from code or via data binding.

In the Designer

Complete the following steps:

1. Click the C1DateTimePicker control once to select it.
2. In the Properties window, set the following properties:
   - Set the **MinDate** property to 01/01/2008.
   - Set the **MaxDate** property to 12/31/2012.
3. Run the program.
4. Click date picker drop-down button to expose the calendar.

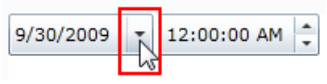


5. Click the forward button to move the calendar ahead until you can no longer go forward. Note that the calendar stops on December 2012.
6. Click the back button to move the calendar back until you can no longer go forward. Note that the calendar stops on January 2008.

In XAML

Complete the following steps:

1. Add **MinDate="2008-01-01"** and **MaxDate="2012-12-31"** to the *<my:C1DateTimePicker>* tag so that the markup resembles the following:
```
<my:C1DateTimePicker Height="26" Name="c1DateTimePicker1" MinDate="2008-01-01" MaxDate="2012-12-31" />
```
2. Run the program.
3. Click date picker drop-down button to expose the calendar.



4. Click the forward button to move the calendar ahead until you can no longer go forward. Note that the calendar stops on December 2012.
5. Click the back button to move the calendar back until you can no longer go forward. Note that the calendar stops on January 2008

In Code

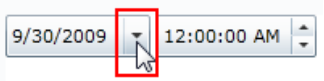
Complete the following:

1. Add **x:Name="C1DateTimePicker1"** to the *<c1datetime: C1DateTimePicker>* tag so that the control will have a unique identifier for you to call in code.
2. Open the **Window1.xaml.cs** page.
3. Place the following code beneath the **InitializeComponent()** method:

Visual Basic
```
'Set the minimum date
C1DateTimePicker1.MinDate = new DateTime(2008, 01, 01)
'Set the maximum date
C1DateTimePicker1.MaxDate = new DateTime(2012, 12, 31)
```

```
C#
//Set the minimum date
c1DateTimePicker1.MinDate = new DateTime(2008, 01, 01);
//Set the maximum date
c1DateTimePicker1.MaxDate = new DateTime(2012, 12, 31);
```

4. Run the program.
5. Click date picker drop-down button to expose the calendar.



6. Click the forward button to move the calendar ahead until you can no longer go forward. Note that the calendar stops on December 2012.
7. Click the back button to move the calendar back until you can no longer go forward. Note that the calendar stops on January 2008.

Specifying the Date and Time

You can specify the time and date of a C1DateTimePicker control by setting the C1DateTimePicker.DateTime property using XAML or code.

> **Note**: Try to avoid avoid setting the **DateTime** property in XAML. Parsing these values from strings is culture specific. If you set a value with your current culture and a user is using different culture, the user can get XamlParseException when loading your site. The best practice is to set these values from code or via data binding.

In XAML

Complete the following steps:

1. Place DateTime="1/17/2010 11:04 AM"to the *<my:C1DateTimePicker>* tag so that the markup resembles the following:

```
<my:C1DateTimePicker DateTime="1/17/2010 11:04 AM"/>
```

2. Run the project and observe that the **C1DateTimePicker** control shows the date and time as 01/17/2010 11:04:00 AM.

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

```
Visual Basic
C1DateTimePicker1.DateTime = New DateTime(2010, 1, 17, 11, 04, 0)
```

C#

```
c1DateTimePicker1.DateTime = new DateTime(2010, 1, 17, 11, 04, 0);
```

3. Run the project and observe that the **C1DateTimePicker** control shows the date and time as 01/17/2010 11:04:00 AM.

This Topic Illustrates the Following:

The result of this topic resembles the following image:

Using C1DateTimePicker Themes (Silverlight)

The C1DateTimePicker control for Silverlight comes equipped with a light blue default theme, but you can also apply six themes (see C1DateTimePicker Theming) to the control. In this topic, you will change the **C1DateTimePicker** control's theme to **C1ThemeRainierOrange**.

In Blend

Complete the following steps:

1. Click the **Assets** tab.
2. In the search bar, enter "C1ThemeRainierOrange". The **C1ThemeRainierOrange** icon appears.
3. Double-click the **C1ThemeRainierOrange** icon to add it to your project.
4. In the search bar, enter "C1DateTimePicker " to search for the **C1DateTimePicker** control.
5. Double-click the **C1DateTimePicker** icon to add the **C1DateTimePicker** control to your project.
6. Under the **Objects and Timeline** tab, select **[C1DateTimePicker]** and use a drag-and-drop operation to place it under **[C1ThemeRainierOrange]**.
7. Run the project.

In Visual Studio

Complete the following steps:

1. Open the **.xaml** page in Visual Studio.
2. Place your cursor between the *<Grid></Grid>* tags.
3. In the Toolbox, double-click the **C1ThemeRainierOrange** icon to declare the theme. Its tags will appear as follows:
   `<my:C1ThemeRainierOrange></my:C1ThemeRainierOrange>`
4. Place your cursor between the *<my:C1ThemeRainierOrange>* and *</my:C1ThemeRainierOrange>* tags.
5. In the Toolbox, double-click the **C1DateTimePicker** icon to add the control to the project. Its tags will appear as children of the *<my:C1ThemeRainierOrange>* tags, causing the markup to resemble the following:

XAML

```
<my:C1ThemeRainierOrange>
    <c1datetime:C1DateTimePicker ></c1datetime:C1DateTimePicker >
</my:C1ThemeRainierOrange>
```

6. Run your project.

This Topic Illustrates the Following:

The following image depicts a C1DateTimePicker control with the **C1ThemeRainierOrange** theme.

C1DatePicker Control Help

Display and choose date information using **DatePicker for WPF and Silverlight**. The C1DatePicker control provides a simple and intuitive UI for selecting date values. Click the C1DatePickers drop-down arrow and select a date in the calendar.

C1DatePicker Quick Start

The following quick start guide is intended to get you up and running with the C1DatePicker control. In this quick start, you'll start in Visual Studio to create a new project, add a **C1DatePicker** control to your application, and customize the **C1DatePicker** control.

Step 1 of 3: Creating an Application with a C1DatePicker Control

In this step, you'll create a WPF or Silverlight application and add a C1DatePicker control to the window.

Complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select **WPF Application** or **Silverlight Application**.
3. Enter a **Name** and **Location** for your project and click **OK** to create the new application.
4. In the Toolbox, double-click the **C1DatePicker** icon to add the **C1DatePicker** control to the application.

You have completed the first step of the **C1DatePicker** quick start. In this step, you created a project and added a **C1DatePicker** control to it. In the next step, you'll customize the control.

Step 2 of 3: Customizing the C1DatePicker

In this step, you will customize the C1DatePicker control.

Select the **C1DatePicker** control and then, in Visual Studio **Properties** window, set the following properties:

- Set the C1DatePicker.SelectedDateFormat property to **Long**. This will change the date format of the control to include the full week day name and month name.
- Click the drop-down arrow next to the C1DatePicker.SelectedDate property and choose **February 14** (or the desired date) from the drop-down calendar.
- Click the drop-down arrow next to the C1DatePicker.ButtonBackground property and select a color from the drop-down color picker.

Now that you've customized the application, you can run the project and observe the run time behaviors of the control.

Step 3 of 3: Running the Application

In the previous two steps, you created a WPF application with a C1DatePicker control and customized the control. In the last step of this quick start, you will run the project and interact with the control.

Press **F5** to run the project. Notice the drop-down arrow is the color you specified in the C1DatePicker.ButtonBackground property. The C1DatePicker.SelectedDate should appear in the display box and it should show the full week day and month names.

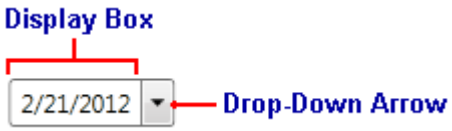
Congratulations – you have completed the quick start. Now that you have finished the quick start, we recommend that you visit the C1DatePicker Task-Based Help topics.

Working with C1DatePicker

The following topics will provide you with an overview of the C1DatePicker control's elements and features.

C1DatePicker Elements

DateTimeEditors for WPF and Silverlight includes the C1DatePicker control, a simple control which provides a date picker that, when clicked at run time, allows you to choose a date from a drop-down calendar. When you add the C1DatePicker control to a XAML window, it exists as a completely functional date picker. By default, the control's interface looks similar to the following image:



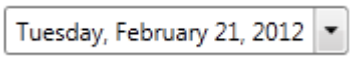
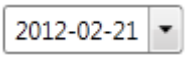
The **C1DatePicker** control consists of the following elements:

- **Display Box**
  The display box presents the selected date. This can be set using the C1DatePicker.SelectedDate property. Users can also input numeric date into the display box at run time. When you enter a numeric value, it will automatically be converted to a date. The control can use the **SelectedDate** property display dates in three edit modes: **Long**, **Short** (default), and **Custom**.
- **Drop-Down Arrow**

Clicking the drop-down arrow of the **C1DatePicker** control at run time allows you to select a date from a drop-down calendar that appears.

Date Formats

You can use the C1DatePicker.SelectedDateFormat property to set the format that the date picker displays. You can set **SelectedDateFormat** property to **Short**, **Long**, or **Custom**. The table below illustrates each date format.

Date Format	Result	Description
Short (default)		The control displays a short date format with numeric values.
Long		The control displays a long date format.
Custom		The control displays the custom format defined in the C1DatePicker.CustomFormat property. **Note:** Use full formats, not abbreviated formats.

C1DatePicker Layout and Appearance

The following topics detail how to customize the C1DatePicker control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases. Themes allow you to customize the appearance of the grid and take advantage of Silverlight's XAML-based styling. You can also use templates to format and layout the control and to customize the control's actions.

C1DatePicker ClearStyle Properties

DateTimeEditors for WPF and Silverlight supports ComponentOne's new ClearStyle technology that allows you to easily change control colors without having to change control templates. By just setting a few color properties you can quickly style the controls.

The following table outlines the brush properties of the C1DatePicker and C1TimeEditor controls:

Brush	Description
Background	Gets or sets the brush of the control's background.
ButtonBackground	Gets or sets the brush of the buttons' background colors.
ButtonForeground	Gets or sets the brush of the buttons' foreground colors.
MouseOverBrush	Gets or sets the System.Windows.Media.Brush used to highlight the buttons when the mouse is hovered over them.
PressedBrush	Gets or sets the System.Windows.Media.Brush used to highlight the buttons when they are clicked on.

You can completely change the appearance of the **C1DatePicker** and **C1TimeEditor** controls by setting a few properties, such as the **ButtonBackground** property, which sets the drop-down button's background color. For example, if you set the **ButtonBackground** property to "#FFC500FF", the controls would appear similar to the following:

C1DatePicker Appearance Properties

The C1DatePicker control includes several properties that allow you to customize the appearance of the controls. You can change the appearance of the text displayed in the controls and customize graphic elements of the controls. The following tables describe some of these appearance properties.

Text Properties

The following properties let you customize the appearance of text in the C1DatePicker control.

Property	Description
FontFamily	Gets or sets the font family of the control. This is a dependency property.
FontSize	Gets or sets the font size. This is a dependency property.
FontStretch	Gets or sets the degree to which a font is condensed or expanded on the screen. This is a dependency property.
FontStyle	Gets or sets the font style. This is a dependency property.
FontWeight	Gets or sets the weight or thickness of the specified font. This is a dependency property.

Color Properties

The following properties let you customize the colors used in the controls.

Property	Description
Background	Gets or sets a brush that describes the background of a control. This is a dependency property.
Foreground	Gets or sets a brush that describes the foreground color. This is a dependency property.

Border Properties

The following properties let you customize the controls' borders.

Property	Description
BorderBrush	Gets or sets a brush that describes the border background of a control. This is a dependency property.
BorderThickness	Gets or sets the border thickness of a control. This is a dependency property.

Size Properties

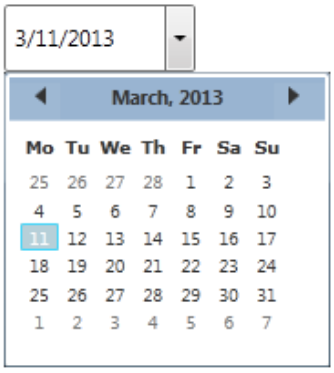
The following properties let you customize the size of the controls.

Property	Description
ActualHeight	Gets the rendered height of this element. This is a dependency property.
ActualWidth	Gets the rendered width of this element. This is a dependency property.
Height	Gets or sets the suggested height of the element. This is a dependency property.
MaxHeight	Gets or sets the maximum height constraint of the element. This is a dependency property.
MaxWidth	Gets or sets the maximum width constraint of the element. This is a dependency property.
MinHeight	Gets or sets the minimum height constraint of the element. This is a dependency property.
MinWidth	Gets or sets the minimum width constraint of the element. This is a dependency property.
Width	Gets or sets the width of the element. This is a dependency property.

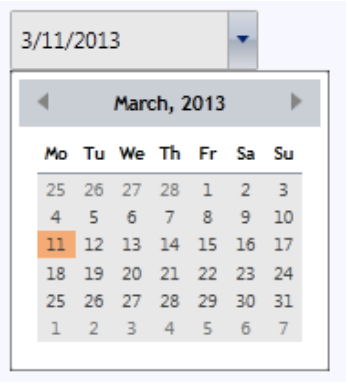
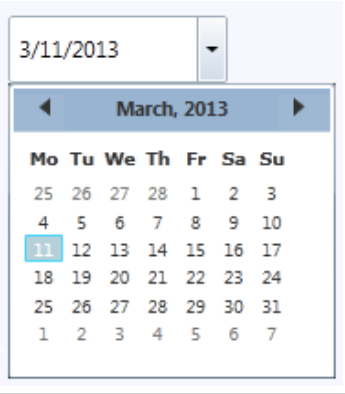
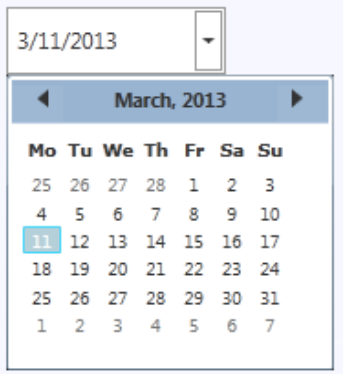
C1DatePicker Theming

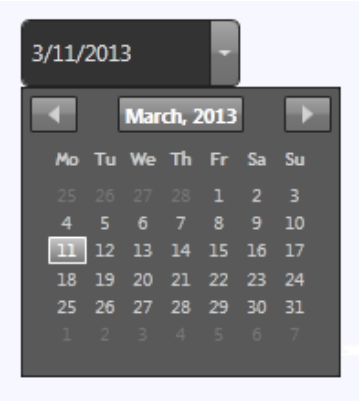
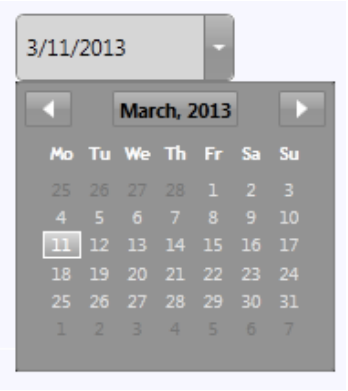
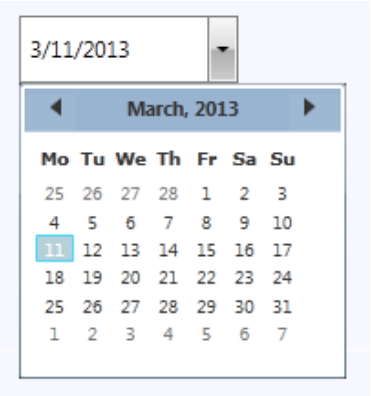
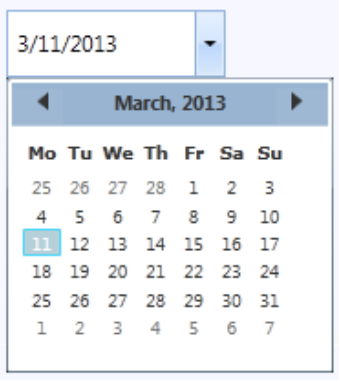
Themes are a collection of image settings that define the look of a control or controls. The benefit of using themes is that you can apply the theme across several controls in the application, thus providing consistency without having to repeat styling tasks.

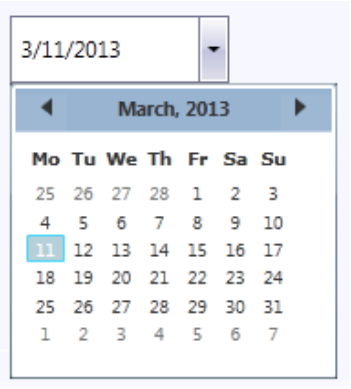
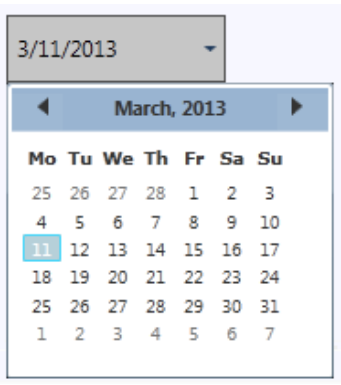
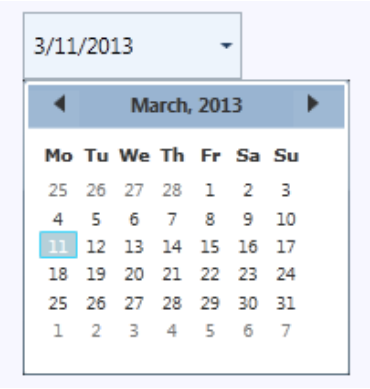
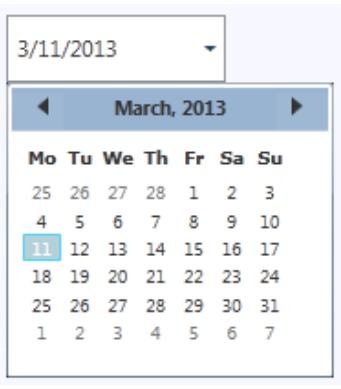
When you add the C1DatePicker control to your project, it appears with the default blue theme, which looks as follows:

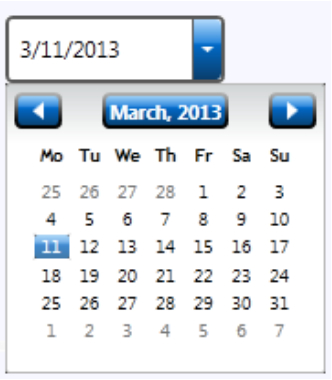
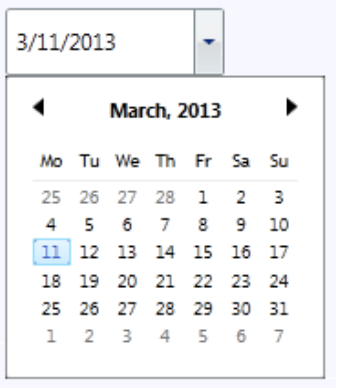


But the **C1DatePicker** control can also be themed with one of our thirteen included WPF themes: BureauBlack, C1Blue, ExpressionDark, ExpressionLight, Office2007Black, Office2007Blue, Office2007Silver, Office2010Black, Office2010Blue, Office2010Silver, ShinyBlue, and WhistlerBlue. The table below provides a sample of each theme:

Full Theme Name	Appearance
BureauBlack	
C1Blue	
Cosmopolitan	

ExpressionDark	
ExpressionLight	
Office2007Black	
Office2007Blue	

Office2007Silver	
Office2010Black	
Office2010Blue	
Office2010Silver	

ShinyBlue	
WhistlerBlue	

To set an element's theme, use the **ApplyTheme** method. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
   Dim theme As New C1ThemeExpressionDark
   ' Using ApplyTheme
   C1Theme.ApplyTheme(LayoutRoot, theme)
```

C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
   {
      C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
      //Using ApplyTheme
      C1Theme.ApplyTheme(LayoutRoot, theme);
   }
```

To apply a theme to the entire application, use the **System.Windows.ResourceDictionary.MergedDictionaries** property. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
   Dim theme As New C1ThemeExpressionDark
```

```
    ' Using Merged Dictionaries
Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme))
End Sub
```

C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
        //Using Merged Dictionaries
Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme));
    }
```

Note that this method works only when you apply a theme for the first time. If you want to switch to another ComponentOne theme, first remove the previous theme from **Application.Current.Resources.MergedDictionaries**.

C1DatePicker Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio and know how to use the C1DatePicker control in general. If you are unfamiliar with the **C1DatePicker** control, please see the C1DatePicker Quick Start first.

Each topic in this section provides a solution for specific tasks using the **C1DatePicker** control.

Each task-based help topic also assumes that you have created a new WPF project.

Allowing Null Values

By default, the C1DatePicker control doesn't allow users to enter null values, but you can force the control to accept a null value by setting the C1DatePicker.AllowNull property to **True**. In this topic, you will learn how to set the **AllowNull** property to **True** in the designer, in XAML, and in code.

In the Designer

Complete the following steps:

1. Click the **C1DatePicker** control once to select it.
2. In the Visual Studio **Properties** window, select the **AllowNull** check box**.**

In XAML

To allow null values, place **AllowNull="True"** within the *<c1:C1DatePicker>* tags so that the markup resembles the following:

```
<c1:C1DatePicker AllowNull="True"/>
```

In Code

Complete the following steps:

1. Open the **MainWindow.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1DatePicker1.AllowNull = True
```

C#

```
c1DatePicker1.AllowNull = true;
```

3. Run the project.

Selecting the Date Format

By default, the C1DatePicker control displays the date in a short format, but it can also display the date in a long or custom format. In this topic, you will learn how to change the date format in the designer, in XAML, and in code.

In the Designer

To change the date format, complete the following steps:

1. Click the **C1DatePicker** control once to select it.
2. In the **Properties** window, click the C1DatePicker.SelectedDateFormat drop-down arrow and select an option from the list. For this example, select **Long**.

In XAML

To change the date format, place **SelectedDateFormat="Long"** within the *<c1:C1DatePicker>* tags so that the markup resembles the following:

```
<c1:C1DatePicker SelectedDateFormat="Long">
```

In Code

To change the date format, complete the following steps:

1. Open the **MainWindow.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1DatePicker1.SelectedDateFormat =
C1.WPF.DateTimeEditors.C1DatePickerFormat.Long
```

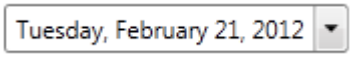
C#

```
c1DateTimePicker1.SelectedDateFormat =
C1.WPF.DateTimeEditors.C1DatePickerFormat.Long;
```

3. Run the project.

This Topic Illustrates the Following:

In this topic, you set the **SelectedDateFormat** to **Long**, which will display the date in a long format. The final result will resemble the following image:



Setting the First Day of the Week

By default, the C1DatePicker control Sunday as the first day of the week in the drop-down calendar, but you can change the starting day if necessary. In this topic, you will learn how to change the first day of the week in the designer, in XAML, and in code.

In the Designer

To change the first day of the week, complete the following steps:

1. Click the **C1DatePicker** control once to select it.
2. In the **Properties** window, click the C1DatePicker.FirstDayOfWeek drop-down arrow and select a day from the list. For this example, select **Monday**.

In XAML

To change the first day of the week, place **FirstDayOfWeek="Monday"** within the *<c1:C1DatePicker>* tags so that the markup resembles the following:

```
<c1:C1DatePicker FirstDayOfWeek="Monday">
```

In Code

To change the date format, complete the following steps:

1. Open the **MainWindow.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

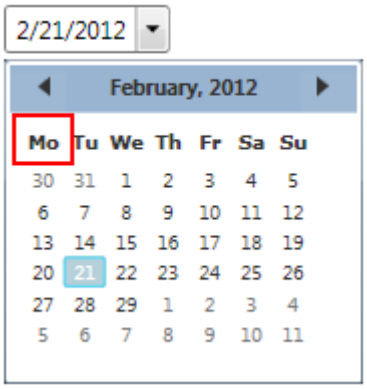
Visual Basic
`C1DatePicker1.FirstDayOfWeek = DayOfWeek.Monday`

C#
`c1DatePicker1.FirstDayOfWeek = DayOfWeek.Monday;`

3. Run the project.

This Topic Illustrates the Following:

In this topic, you set the **FirstDayOfWeek** property to **Monday** so that Monday is the first day of the week in the drop-down calendar. The final result will resemble the following image:



Setting the Calendar Start and End Date

You can change the dates that appear in the drop-down calendar by setting the C1DatePicker.DisplayDateStart and C1DatePicker.DisplayDateEnd properties. In this topic, you will learn how to change the start and end dates in the designer, in XAML, and in code.

In the Designer

To change the dates that appear in the calendar, complete the following steps:

1. Click the C1DatePicker control once to select it.
2. In the **Properties** window, click the **DisplayDateStart** drop-down arrow and select a day from the list. For this example, select **2/5/2012**.
3. In the **Properties** window, click the **DisplayDateEnd** drop-down arrow and select a day from the list. For this example, select **2/25/2012**.

In XAML

To specify the first and last day of the calendar, place **DisplayDateStart="02/05/2012"** and **DisplayDateEnd="02/25/2012"** within the *<c1:C1DatePicker>* tags so that the markup resembles the following:

XAML

```
<c1:C1DatePicker Name="C1DatePicker1" DisplayDateEnd="02/25/2012"
DisplayDateStart="02/05/2012">
```

In Code

To change the dates that appear in the calendar, complete the following steps:

1. Open the MainWindow.xaml.cs page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
Dim dateStringStart As String = "02/05/2012"
Dim dateStringEnd As String = "02/25/2012"
```

```
C1DatePicker1.DisplayDateStart = DateTime.Parse(dateStringStart)
C1DatePicker1.DisplayDateEnd = DateTime.Parse(dateStringEnd)
```

C#

```
string dateStringStart = "02/05/2012";
string dateStringEnd = "02/25/2012";

C1DatePicker1.DisplayDateStart = DateTime.Parse(dateStringStart);
C1DatePicker1.DisplayDateEnd = DateTime.Parse(dateStringEnd);
```

3. Run the project.

This Topic Illustrates the Following:

In this topic, you set the **DisplayDateStart** and **DisplayDateEnd** properties to determine the dates that appear in the drop-down calendar. The final result will resemble the following image:

Using C1DatePicker Themes (Silverlight)

The C1DatePicker control allows you to apply six themes to the control. In this topic, you will change the **C1DatePicker** control's theme to **C1ThemeRainierOrange**. For more information on available themes, see C1DatePicker Theming.

In Blend

Complete the following steps:

1. Click the **Assets** tab.
2. In the search bar, enter "C1ThemeRainierOrange".
   The **C1ThemeRainierOrange** icon appears.
3. Double-click the **C1ThemeRainierOrange** icon to add it to your project.
4. In the search bar, enter "C1DatePicker" to search for the **C1DatePicker** control.
5. Double-click the **C1DatePicker** icon to add the **C1DatePicker** control to your project.
6. Under the **Objects and Timeline** tab, select **[C1DatePicker]** and use a drag-and-drop operation to place it under **[C1ThemeRainierOrange]**.

7. Run the project.

In Visual Studio

Complete the following steps:

1. Open the .xaml page in Visual Studio.
2. Place your cursor between the <*Grid*><*/Grid*> tags.
3. In the Toolbox, double-click the **C1ThemeRainierOrange** icon to declare the theme. Its tags will appear as follows:
   `<c1:C1ThemeRainierOrange></c1:C1ThemeRainierOrange>`
4. Place your cursor between the <*c1:C1ThemeRainierOrange*> and <*/c1:C1ThemeRainierOrange*> tags.
5. In the Toolbox, double-click the **C1DatePicker** icon to add the control to the project. Its tags will appear as children of the <*c1:C1ThemeRainierOrange*> tags, causing the markup to resemble the following:

   XAML

   ```
   <c1:C1ThemeRainierOrange>
      <c1:C1DatePicker/>
   </c1:C1ThemeRainierOrange>
   ```

6. Run your project.

C1TimeEditor Control Help

Exchange time information with an end-user using **TimePicker for WPF and Silverlight**. It provides a simple interface for selecting time values. The time can be selected by using the spin buttons, keyboard arrows, or by typing in fields.

C1TimeEditor Quick Start

The following quick start guide is intended to get you up and running with the C1TimeEditor control. In this quick start, you'll start in Visual Studio to create a new project, add a **C1TimeEditor** control to your application, and customize the **C1TimeEditor** control.

Step 1 of 3: Creating an Application with a C1TimeEditor Control

In this step, you'll begin in Visual Studio to create a WPF or Silverlight application using the C1TimeEditor control.

If creating a WPF project, complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select **WPF Application**.
3. Enter a **Name** and **Location** for your project and click **OK** to create the new application.
4. In the Toolbox, double-click the **C1TimeEditor** icon to add the **C1TimeEditor** control to the WPF application.

If creating a Silverlight project, complete the following steps:

1. In Expression Blend, select **File | New Project**.
2. In the **New Project** dialog box, select the Silverlight project type in the left pane and, in the right-pane, select **Silverlight Application + Website**.
3. Enter a **Name** and **Location** for your project and click **OK**. Blend creates a new application, which opens with the **MainPage.xaml** file displayed in Design view.
4. Add the **C1TimeEditor** control to your project by completing the following steps:
    a. On the menu, select **Window | Assets** to open the **Assets** tab.
    b. Under the **Assets** tab, enter "C1TimeEditor" into the search bar.
    c. The **C1TimeEditor** control's icon appears.
    d. Double-click the **C1TimeEditor** icon to add the control to your project.

You have completed the first step of the **C1TimeEditor** quick start. In this step, you created a project and added a **C1TimeEditor** control to it. In the next step, you'll customize the control.

Step 2 of 3: Customizing the C1TimeEditor

In this step, you will customize the C1TimeEditor control.

In the WPF project, select the **C1TimeEditor** control and then, in Visual Studio **Properties** window, set the following properties:

- Set the C1TimeEditor.Format property to **ShortTime**. This will change the time format of the control so that it

shows only hours and minutes.
- Set the C1TimeEditor.Increment property to "01:00:00". This will cause the value of the control to change by one hour each time a user clicks the spin button.
- Set the C1TimeEditor.Interval property to "1000". This will cause the control to hesitate for one second before changing the value of the control.
- Set the C1TimeEditor.Value property to "17:00:00".

In the Silverlight project in Blend, select the **C1TimeEditor** control and then, in **Properties** panel, set the following properties:

1. Set the **C1TimeEditor.Format** property to **ShortTime**. This will change the time format of the control so that it shows only hours and minutes.
2. Set the **C1TimeEditor.Increment** property to "01:00:00". This will cause the value of the control to change by one hour each time a user clicks the spin button.
3. Set the **C1TimeEditor.Interval** property to "1000". This will cause the control to hesitate for one second before changing the value of the control.
4. Set the current time to **5:00 p.m**. by completing the following steps:
    a. In the XAML editor, add **x:Name="C1TimeEditor1"** to the *<c1datetime:C1TimeEditor>* tag so that the control will have a unique identifier for you to call in code.
    b. Open the MainPage.xaml.cs page.
    c. Place the following code beneath the **InitializeComponent()** method:

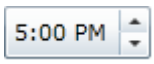
Now that you've customized the application, you can run the project and observe the run time behaviors of the control.

Step 3 of 3: Running the Application

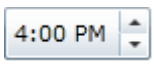
In the previous two steps, you created a WPF or Silverlight application with a C1TimeEditor control and customized the control. In the last step of this quick start, you will run the project and interact with the control.

Complete the following steps:

1. Press **F5** (or **Project | Run** in Blend) to run the project. Observe that it loads with a time value of 5:00 p.m. and that the control only shows hours and minutes.



2. Click the decrease time button and observe that time value decreases by one hour to 4:00 p.m.

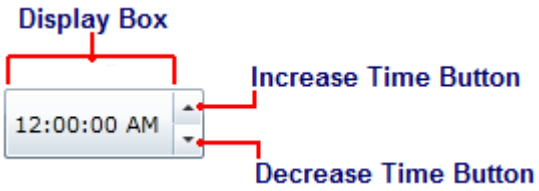


3. Click and hold the increase time button so that the control will spin through values. Observe that the control waits one second between value changes.

Congratulations – you have completed the quick start. Now that you have finished the quick start, we recommend that you visit the Working with C1TimeEditor or C1TimeEditor Task-Based Help topics.

C1TimeEditor Elements

DateTimeEditors for WPF and Silverlight includes the C1TimeEditor control, a simple control which provides a time picker that can show short time, long time, and time spans. When you add the C1TimeEditor control to a XAML window, it exists as a completely functional time picker. By default, the control's interface looks similar to the following image:



The **C1TimeEditor** control consists of the following elements:

- **Display Box**
  The display box presents the selected time. This can be set using the C1TimeEditor.Value property. Users can also input numeric date into the display box. When you enter a numeric value, it will automatically be converted into time. The control can display time in three edit modes: **LongTime** (default), **ShortTime**, and **TimeSpan**.
- **Increase Time Button**
  The increase time button allows you to increase the time displayed in the time picker. Clicking the increase button will increase the time by one minute unless you have specified another interval.
- **Decrease Time Button**
  The decrease time button allows you to decrease the time displayed in the time picker. Clicking the decrease button will decrease the time by one minute unless you have specified another interval.

Spin Interval

There are two ways that users can increase or decrease values using the spin button: they can either repeatedly click one of the buttons to increase or decrease the time at their own pace, or they can hold down the decrease time button or increase time button while time increases or decreases at the speed of program-specified intervals. You can specify the interval by setting the C1TimeEditor.Interval property.

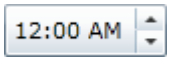
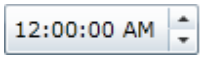
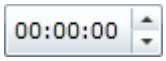
By default, the **Interval** property is set to 33 milliseconds, which allows users to scroll through time values at faster rates. You can slow that scrolling time down by specifying a higher number, such as 500 milliseconds (one-half of a second), or speed it up by specifying a lower number, such as 10 milliseconds (one-hundredth of a second). You cannot set the **Interval** to "0".

Value Increment

Each time a user clicks the increase time or decrease time spin buttons, the value of the control increases or decreases by a program-specified increment. By default, this increment is 00:01:00, or one minute. You can increase or decrease this increment by setting the C1TimeEditor.Increment property. The **Increment** property will take any value between 00:00:00 (which will disable the spin buttons) and 23:59:59.

Time Formats

You can use the C1TimeEditor.Format property to set the format that the date picker displays. You can set **Format** property to **ShortTime**, **LongTime**, or **TimeSpan**. The table below illustrates each date formats.

Time Format	Result	Description
ShortTime		The control displays a short time format that excludes seconds.
LongTime (default)		The control displays a long time format that includes seconds.
TimeSpan		The control displays a time span and removes the a.m./p.m. designators.

C1TimeEditor for WPF Layout and Appearance

The following topics detail how to customize the C1TimeEditor control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases. Themes allow you to customize the appearance of the grid and take advantage of Silverlight's XAML-based styling. You can also use templates to format and layout the control and to customize the control's actions.

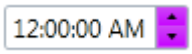
C1TimeEditor ClearStyle Properties

C1TimeEditor supports ComponentOne's new ClearStyle technology that allows you to easily change control colors without having to change control templates. By just setting a few color properties you can quickly style the controls.

The following table outlines the brush properties of the C1TimeEditor control:

Brush	Description
Background	Gets or sets the brush of the control's background.
ButtonBackground	Gets or sets the brush of the buttons' background colors.
ButtonForeground	Gets or sets the brush of the buttons' foreground colors.
MouseOverBrush	Gets or sets the System.Windows.Media.Brush used to highlight the buttons when the mouse is hovered over them.
PressedBrush	Gets or sets the System.Windows.Media.Brush used to highlight the buttons when they are clicked on.

You can completely change the appearance of the **C1TimeEditor** control by setting a few properties, such as the **ButtonBackground** property, which sets the drop-down button's background color. For example, if you set the **ButtonBackground** property to "#FFC500FF", the controls would appear similar to the following:



C1TimeEditor Appearance Properties

The C1TimeEditor control includes several properties that allow you to customize the appearance of the controls. You can change the appearance of the text displayed in the controls and customize graphic elements of the controls. The following tables describe some of these appearance properties.

Text Properties

The following properties let you customize the appearance of text in the C1TimeEditor control.

Property	Description
FontFamily	Gets or sets the font family of the control. This is a dependency property.
FontSize	Gets or sets the font size. This is a dependency property.
FontStretch	Gets or sets the degree to which a font is condensed or expanded on the screen. This is a dependency property.
FontStyle	Gets or sets the font style. This is a dependency property.
FontWeight	Gets or sets the weight or thickness of the specified font. This is a dependency property.

Color Properties

The following properties let you customize the colors used in the controls.

Property	Description
Background	Gets or sets a brush that describes the background of a control. This is a dependency property.
Foreground	Gets or sets a brush that describes the foreground color. This is a dependency property.

Border Properties

The following properties let you customize the controls' borders.

Property	Description
BorderBrush	Gets or sets a brush that describes the border background of a control. This is a dependency property.
BorderThickness	Gets or sets the border thickness of a control. This is a dependency property.

Size Properties

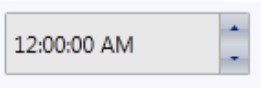
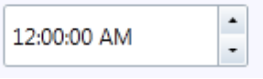
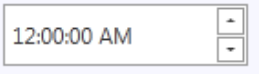
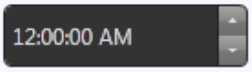
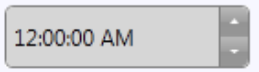
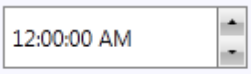
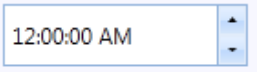
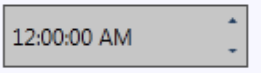
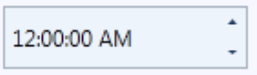
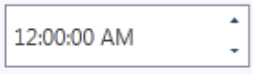
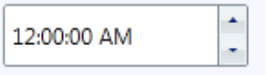
The following properties let you customize the size of the controls.

Property	Description
ActualHeight	Gets the rendered height of this element. This is a dependency property.
ActualWidth	Gets the rendered width of this element. This is a dependency property.
Height	Gets or sets the suggested height of the element. This is a dependency property.
MaxHeight	Gets or sets the maximum height constraint of the element. This is a dependency property.
MaxWidth	Gets or sets the maximum width constraint of the element. This is a dependency property.
MinHeight	Gets or sets the minimum height constraint of the element. This is a dependency property.
MinWidth	Gets or sets the minimum width constraint of the element. This is a dependency property.
Width	Gets or sets the width of the element. This is a dependency property.

C1TimeEditor Theming

TimeEditor for WPF and Silverlight incorporates several themes that allow you to customize the appearance of your controls. When you first add one of the **C1TimeEditors** controls to the page, they appear similar to the following image:

But the C1TimeEditor control can also be themed with one of our thirteen included WPF themes: *BureauBlack, C1Blue, ExpressionDark, ExpressionLight, Office2007Black, Office2007Blue, Office2007Silver, Office2010Black, Office2010Blue, Office2010Silver, ShinyBlue,* and *WhistlerBlue*. The table below provides a sample of each theme:

Theme Name	Theme Preview
BureauBlack	
C1Blue	
Cosmopolitan	
ExpressionDark	
ExpressionLight	
Office2007Black	
Office2007Blue	
Office2007Silver	
Office2010Black	
Office2010Blue	
Office2010Silver	
ShinyBlue	
WhistlerBlue	

To set an element's theme, use the **ApplyTheme** method. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark
    ' Using ApplyTheme
    C1Theme.ApplyTheme(LayoutRoot, theme)
```

Example Title

```
private void Window_Loaded(object sender, RoutedEventArgs e)
```

```
    {
        C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
        //Using ApplyTheme
        C1Theme.ApplyTheme(LayoutRoot, theme);
    }
```

To apply a theme to the entire application, use the **System.Windows.ResourceDictionary.MergedDictionaries** property. First add a reference to the theme assembly to your project, and then set the theme in code, like this:

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark
    ' Using Merged Dictionaries
Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme))
End Sub
```

C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
    {
        C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
        //Using Merged Dictionaries

Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme));
    }
```

Note that this method works only when you apply a theme for the first time. If you want to switch to another ComponentOne theme, first remove the previous theme from **Application.Current.Resources.MergedDictionaries**.

C1TimeEditor Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio and know how to use the C1TimeEditor control in general. If you are unfamiliar with the **C1TimeEditor** control, please see the C1TimeEditor Quick Start first.

Each topic in this section provides a solution for specific tasks using the **C1TimeEditor** control.

Each task-based help topic also assumes that you have created a new WPF project.

Allowing Null Values

By default, the C1TimeEditor control doesn't allow users to enter null values, but you can force the control to accept a null value by setting the C1TimeEditor.AllowNull property to **True**. In this topic, you will learn how to set the **AllowNull** property to **True** in the designer, in XAML, and in code.

In the Designer

Complete the following steps:

1. Click the **C1TimeEditor** control once to select it.
2. In the **Properties** window, select the **AllowNull** check box.

In XAML

To allow null values, place **AllowNull="True"** to the *<my:C1TimeEditor>* tag so that the markup resembles the following:

```
<my:C1TimeEditor AllowNull="True"/>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1TimeEditor1.AllowNull = True
```

C#

```
c1TimeEditor1.AllowNull = true;
```

3. Run the project.

Removing the Spin Buttons

You can remove the C1TimeEditor control's spin buttons by setting the C1TimeEditor.ShowButtons property to **False**. In this topic, you will learn how to set the **ShowButtons** property to **False** in the designer, in XAML, and in code.

In the Designer

Complete the following steps:

1. Click the **C1TimeEditor** control once to select it.
2. In the **Properties** window, clear the **ShowButtons** check box.

In XAML

To remove the spin buttons, place **ShowButtons="False"** to the *<my:C1TimeEditor>* tag so that the markup resembles the following:

```
<my:C1TimeEditor ShowButtons="False"/>
```

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1TimeEditor1.ShowButtons = False
```

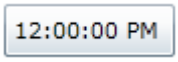
C#

```
c1TimeEditor1.ShowButtons = false;
```

3. Run the project.

This Topic Illustrates the Following:

The following image depicts a **C1TimeEditor** control with its spin buttons removed.



Selecting the Time Format

By default, the C1TimeEditor control displays the time in a long format that includes seconds, but it can also display time in a shorter format or into a time span format. In this topic, you will learn how to change the time format in the designer, in XAML, and in code.

In the Designer

To change the time format, complete the following steps:

1. Click the **C1TimeEditor** control once to select it.
2. In the **Properties** window, click the **Format** drop-down arrow and select a mode from the list. For this example, select **ShortTime**.

In XAML

To change the time format, place **Format="ShortTime"** to the *<my:C1TimeEditor>* tag so that the markup resembles the following:

```
<my:C1TimeEditor Format="ShortTime">
```

In Code

To change the time format, complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Import the following namespace:

Visual Basic

```
Imports C1.WPF.DateTimeEditors
```

C#

```
using C1.WPF.DateTimeEditors;
```

3. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1TimeEditor1.Format = C1TimeEditorFormat.ShortTime
```

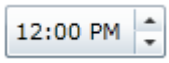
C#

```
c1TimeEditor1.Format = C1TimeEditorFormat.ShortTime;
```

4. Run the project.

   This Topic Illustrates the Following:

   In this topic, you set the Format to **ShortTime**, which provides a shortened time display. The final result will resemble the following image:



Setting the Spin Interval

By default, the C1TimeEditor.Interval property is set to 33 milliseconds, which allows users to scroll through the time values at faster rates. In this topic, you will specify a longer interval between value changes by setting the **Interval** property to 1000 milliseconds. For more information on spin intervals, visit the Spin Interval topic.

In the Designer

Complete the following steps:

1. Click the C1TimeEditor control once to select it.
2. In the **Properties** window, locate the **Interval** property and enter "1000" into its text box.
3. Run the project and then click and hold the increase time button. Observe that the value only increases once a second.

In XAML

Complete the following steps:

1. Add **Interval="1000"** to the *<my:C1TimeEditor>* tag so that the markup resembles the following:
   ```
   <my:C1TimeEditor Interval="1000"/>
   ```
2. Run the project and then click and hold the increase time button. Observe that the value only increases once a second.

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1TimeEditor1.Interval = 1000
```

C#

```
c1TimeEditor1.Interval = 1000;
```

3. Run the project and then click and hold the increase time button. Observe that the value only increases once a second.

Setting the Value Increment

By default, the time on a C1TimeEditor control is set to move in one minute increments. You can change this by setting the C1TimeEditor.Increment property to whatever time increment you specify. In this topic, you will set the time increment on the **C1TimeEditor** control to one hour and thirty minutes. For more information about time increments, visit the Value Increment topic.

In the Designer

Complete the following steps:

1. Click the **C1TimeEditor** control once to select it.
2. In the **Properties** window, locate the **Increment** property and enter **"01:30:00"** into its text box.
3. Run the project and click the increase time button. Observe that time jumps ahead by one hour and thirty minutes.

In XAML

Complete the following steps:

1. Add **Increment="01:30:00"** to the *<my:C1TimeEditor>* tag so that the markup resembles the following:
   `<my:C1TimeEditor Increment="01:30:00"/>`
2. Run the project and click the increase time button. Observe that time jumps ahead by one hour and thirty minutes.

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1TimeEditor1.Increment = New TimeSpan(01, 30, 00)
```

C#

```
c1TimeEditor1.Increment = new TimeSpan(01,30,00);
```

3. Run the project and click the increase time button . Observe that time jumps ahead by one hour and thirty minutes.

Specifying the Current Time

You can specify the current time of a C1TimeEditor control by setting the C1TimeEditor.Value property in the designer, in XAML, and in code.

> **Note**: Try to avoid setting the **Value** property in XAML as a string value. Parsing these values from strings is culture specific. If you set a value with your current culture and a user is using different culture, the user can get XamlParseException when loading your site. The best practice is to set these values from code or via data binding.

In the Designer

Complete the following steps:

1. Click the **C1TimeEditor** control once to select it.
2. In the **Properties** window, set the **Value** property to **" 07:00:00"**. The control reflects a time of 7:00:00 a.m.

In XAML

To specify the current time, place **Value="07:00:00"** to the *<my:C1TimeEditor>* tag so that the markup resembles the following:

```
<my:C1TimeEditor Value="07:00:00"/>
```

The control reflects a time of 7:00:00 a.m.

In Code

Complete the following steps:

1. Open the **Window1.xaml.cs** page.
2. Place the following code beneath the **InitializeComponent()** method:

Visual Basic

```
C1TimeEditor1.Value = New TimeSpan(7, 0, 0)
```

C#

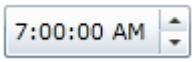
```
c1TimeEditor1.Value = new TimeSpan(7,0,0);
```

3. Run the project and observe that control reflects a time of 7:00:00 a.m.

This Topic Illustrates the Following:

By following the steps in this topic, you have changed the time on the **C1TimeEditor** control to 7:00:00 a.m. The result

will resemble the following:



Working with Time Spans

You can modify a C1TimeEditor control so that it will display a time span. In this tutorial, you will create a **C1TimeEditor** control that represents a time span between 5:00 and 10:00. You will also write code for the project that sets the starting value to 7:00 a.m.

Complete the following steps:

1. Click the **C1TimeEditor** control once to select it.
2. In the **Properties** window, complete the following steps:
    - Set the Format property to **TimeSpan**.
    - Set the C1TimeEditor.Maximum property to a value, such as "10:00:00".
    - Set the C1TimeEditor.Minimum property to a value, such as "05:00:00".
    - Set the C1TimeEditor.Value property to a value, such as "07:00:00".
3. Run the project and observe that the control loads with a time of 07:00:00 a.m.
4. Click the increase time button until you can go no further. It will stop at 10:00:00.
5. Click the decrease time button until you can go no further. It will stop at 05:00:00.

Using C1TimeEditor Themes

The C1TimeEditor control comes equipped with a light blue default theme, but you can also apply six themes (see C1TimeEditor Theming) to the control. In this topic, you will change the **C1TimeEditor** control's theme to **C1ThemeRainierOrange**.

In Blend

Complete the following steps:

1. Click the **Assets** tab.
2. In the search bar, enter "C1ThemeRainierOrange". The **C1ThemeRainierOrange** icon appears.
3. Double-click the **C1ThemeRainierOrange** icon to add it to your project.
4. In the search bar, enter "C1TimeEditor" to search for the **C1TimeEditor** control.
5. Double-click the **C1TimeEditor** icon to add the **C1TimeEditor** control to your project.
6. Under the **Objects and Timeline** tab, select **[C1TimeEditor]** and use a drag-and-drop operation to place it under **[C1ThemeRainierOrange]**.
7. Run the project.

In Visual Studio

Complete the following steps:

1. Open the **.xaml** page in Visual Studio.
2. Place your cursor between the *<Grid></Grid>* tags.
3. In the Toolbox, double-click the **C1ThemeRainierOrange** icon to declare the theme. Its tags will appear as follows:

```
<my:C1ThemeRainierOrange></my:C1ThemeRainierOrange>
```

4. Place your cursor between the *<my:C1ThemeRainierOrange>* and *</my:C1ThemeRainierOrange>* tags.
5. In the Toolbox, double-click the C1TimeEditor icon to add the control to the project. Its tags will appear as children of the *<my:C1ThemeRainierOrange>* tags, causing the markup to resemble the following:

XAML

```
<my:C1ThemeRainierOrange>
     <c1datetime:C1TimeEditor></c1datetime:C1TimeEditor>
</my:C1ThemeRainierOrange>
```

6. Run your project.

This Topic Illustrates the Following:

The following image depicts a **C1TimeEditor** control with the **C1ThemeRainierOrange** theme.