ComponentOne

Calendar for Windows Phone

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

Internet: **info@ComponentOne.com**

Web site: **http://www.componentone.com**

Sales

E-mail: sales@componentone.com

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

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

Table of Contents

ComponentOne Calendar for Windows Phone Overview

Deliver your own Metro-style calendars with ComponentOne Calendar™ for Windows Phone. Get gesture-based navigation, date range selection and a highly customizable styling model for quick and simple app development.

For a list of the latest features added to **ComponentOne Studio for Windows Phone**, visit What's New in Studio for Windows Phone.

Installing Studio for Windows Phone

The following sections provide helpful information on installing **ComponentOne Studio for Windows Phone**.

Studio for Windows Phone Setup Files

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

Bin	Contains copies of ComponentOne binaries (DLLs, EXEs, design-time assemblies).
Help	Contains documentation for all Studio components and other useful resources including XAML files.

Samples

Samples for the product are installed in the **ComponentOne Samples** folder by default.

Windows Vista and Windows 7 path: C:\Users\<username>\Documents\ComponentOne Samples\Studio for Windows Phone

See the Studio for Windows Phone Samples topic for more information about each sample.

System Requirements

System requirements for **ComponentOne Studio for Windows Phone** include the following:

- Microsoft Visual Studio 2010
- Microsoft Expression Blend 4
- Microsoft Windows Phone SDK
- Visual Basic for Windows Phone Developer Tools (if using Visual Basic)

Installing Demonstration Versions

If you wish to try **ComponentOne Studio for Windows Phone** and do not have a serial number, follow the steps through the installation wizard and use the default serial number.

The only difference between unregistered (demonstration) and registered (purchased) versions of our products is that the registered version will stamp every application you compile so a ComponentOne banner will not appear when your users run the applications.

Uninstalling Studio for Windows Phone

To uninstall **ComponentOne Studio for Windows Phone**:

1. Open the **Control Panel** and select **Programs and Features**.
2. Select **ComponentOne Studio for Windows Phone** and click the **Remove** button.
3. Click **Yes** to remove the program.

Technical Support

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

Some methods for obtaining technical support include:

- **Online Support via HelpCentral**
  ComponentOne HelpCentral provides customers with a comprehensive set of technical resources in the form of FAQs, samples, Version Release History, Articles, searchable Knowledge Base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**
  This online support service provides you with direct access to our Technical Support staff via an online incident submission form. When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.

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

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

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

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

Redistributable Files

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

- C1.Phone.dll
- C1.Phone.Chart.dll
- C1.Phone.Extended.dll
- C1.Phone.Gauge.dll
- C1.Phone.Imaging.dll
- C1.Phone.Maps.dll
- C1.Phone.Pdf.dll
- C1.Phone.PdfViewer.sll
- C1,Phone.RichTextBox.dll
- C1.Phone.RichTextBox.ApplicationBar.dll
- C1.Phone.RichTextBox.RtfFilter.dll

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

About This Documentation

Acknowledgements

Microsoft, Windows, Windows Vista, and Visual Studio, and Silverlight, are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Firefox is a registered trademark of the Mozilla Foundation. Safari is a trademark of Apple Inc., registered in the U.S. and other countries.

ComponentOne

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

Corporate Headquarters

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

http://www.componentone.com

ComponentOne Doc-To-Help

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

Assemblies and Controls

ComponentOne Studio for Windows Phone includes several assemblies. This increases the granularity of the studio and helps to decrease the size of applications that do not use all the controls available in the studio.

Studio for Windows Phone Samples

If you just installed **ComponentOne Studio for Windows Phone**, open Visual Studio 2010 and load the **Samples.sln** solution located in the **C:\Documents and Settings\<username>\My Documents\ComponentOne Samples\Studio for Windows Phone** or **C:\Users\<username>\Documents\ComponentOne Samples\Studio for Windows Phone** folder. This solution contains all the samples that ship with this release. Each sample has a readme.txt file that describes it and two projects named as follows:

<SampleName>	Windows Phone project (client-side project)

Introduction to Windows Phone

The following topics detail information about getting started with Windows Phone, including Windows Phone resources, and general information about templates and deploying Windows Phone files.

Windows Phone Resources

This help file focuses on **ComponentOne Studio for Windows Phone**. For general help on getting started with Windows Phone, we recommend the following resources:

- http://www.silverlight.net

  The official Silverlight site, with many links to downloads, samples, tutorials, and more for developing applications for Windows Phone.

- http://create.msdn.com/en-us/education/quickstarts

  Windows Phone development Quick Starts on App Hub. These tutorials are great if you're new (or even not-so-new) to Windows Phone development.

- http://www.silverlight.net/learn/tutorials/jesse-liberty/windows-phone-7-tutorials/

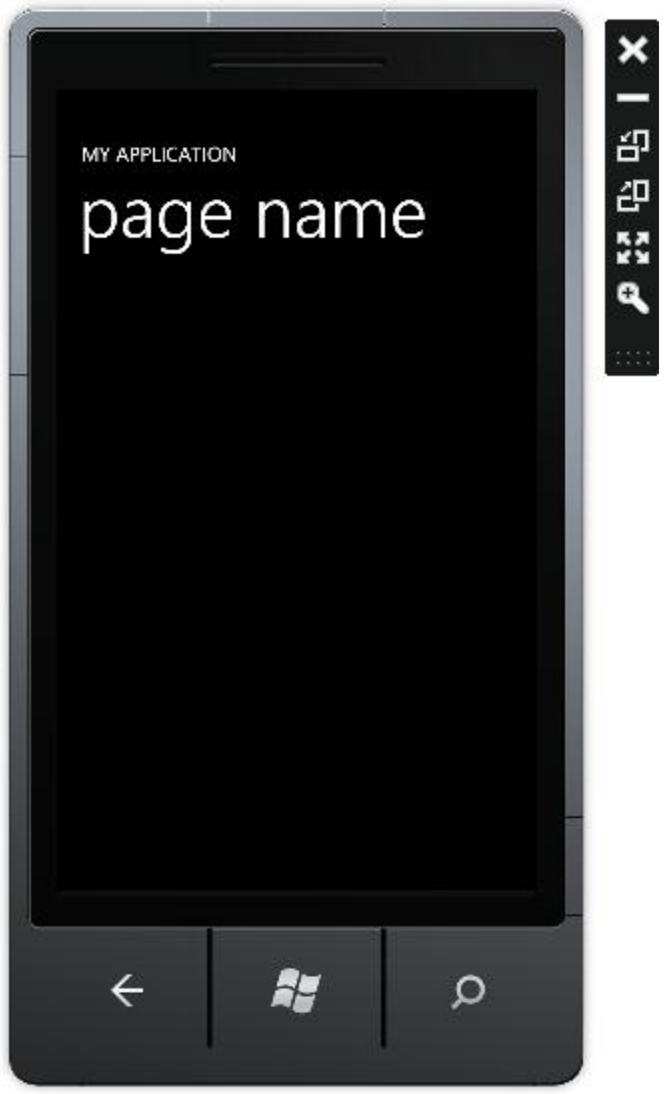
  Jesse Liberty's Windows Phone 7 Tutorials. Jesse Liberties Silverlight tutorials are also a great resource.

- http://msdn.microsoft.com/en-us/library/ff402535(v=vs.92).aspx

  Windows Phone Development documentation on MSDN. This page includes links to the fundamentals as well as more in-depth how-tos.

Windows Phone 7 Emulator

You can use the Windows Phone 7 Emulator to run Windows Phone 7 applications on your development machine. If you've installed the Windows Phone Development Tools, you should be able to access the emulator by clicking the **Start** button and then **All Programs** │ **Windows Phone Developer** │ **Tools Windows Phone 7 Emulator**. When you debug a Windows Phone application, it should appear displayed in the Windows Phone 7 Emulator application by default.

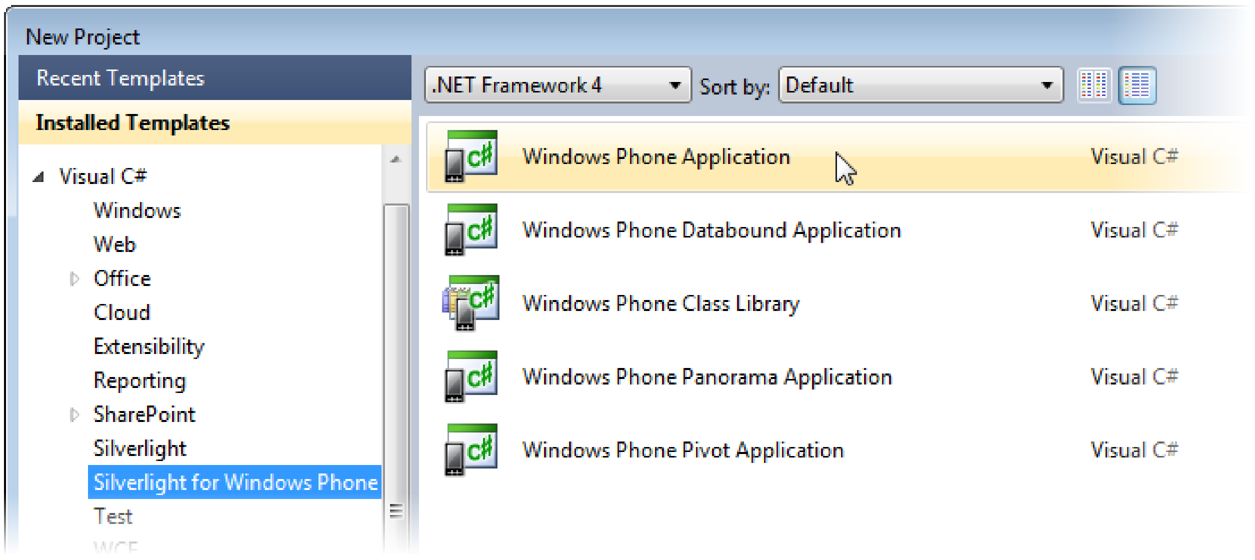


The menu below the screen includes the **Back**, **Start**, and **Search** buttons. The menu displayed at the right of the screen in the image above, includes the **Close**, **Minimize**, **Rotate Right**, **Rotate Left**, **Expand**, and **Zoom** buttons. While the emulator is not a perfect representation of the touch interaction possible with the Windows Phone, it works well to test and debug Windows Phone applications.

Creating a New Windows Phone Project

The following topic details how to create a new Windows Phone project in Microsoft Visual Studio 2010. Complete the following steps to create a new Windows Phone project in Microsoft Visual Studio 2010:

1. Select **File | New | Project** to open the **New Project** dialog box in Visual Studio 2010.
2. In the **Project types** pane, expand the **Visual Basic** or **Visual C#** node and select **Silverlight for Windows Phone**.
3. Choose **Windows Phone Application** in the **Templates** pane.


New Project
Recent Templates
Installed Templates
.NET Framework 4
Sort by: Default
Visual C#
Windows
Web
Office
Cloud
Extensibility
Reporting
SharePoint
Silverlight
Silverlight for Windows Phone
Test
Windows Phone Application
Windows Phone Databound Application
Windows Phone Class Library
Windows Phone Panorama Application
Windows Phone Pivot Application
Visual C#

4. Name the project, specify a location for the project, and click **OK**.

   A new project will be created with the name you specified.

Adding Studio for Windows Phone Components to a Project

This topic details how you can add the **Studio for Windows Phone** components to the Toolbox, add controls to a project, and to add references to the **Studio for Windows** Phone assemblies. When the **Studio for Windows Phone** controls are in the Visual Studio Toolbox, you can easily add them to a project at design time.

Manually Adding Studio for Windows Phone Controls to the Toolbox

To add the **ComponentOne Studio for Windows Phone** controls to the Toolbox, complete the following steps:

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu, if necessary) and right-click the Toolbox to open its context menu.
2. To make Studio for Windows Phone components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **C1Phone**, for example.
3. Select **File | New | Project** to open the **New Project** dialog box in Visual Studio 2010.
4. In the **Project types** pane, expand the **Visual Basic** or **Visual C#** node and select **Silverlight for Windows Phone**.
5. Right-click the tab where the component is to appear and select **Choose Items** from the context menu.
6. The **Choose Toolbox Items** dialog box opens.
7. In the dialog box, select the **Windows Phone Components** tab. Sort the list by **Namespace** (click the **Namespace** column header) and check the check boxes for all components belonging to the C1.Phone, C1.Phone.Chart, and C1.Phone.Extended namespaces. Note that there may be more than one component for each namespace.
8. Name the project, specify a location for the project, and click **OK**.

   A new project will be created with the name you specified.

Adding Studio for Windows Phone Controls to the Application

To add **ComponentOne Studio for Windows Phone** controls to the application, complete the following steps:

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

Adding Assembly References in Code

To add a reference to the **ComponentOne Studio for Windows Phone** assemblies, complete the following steps:

3. Select the **Add Reference** option from the **Project** menu of your project.
4. Select the **ComponentOne Studio for Windows Phone** assemblies from the list on the **.NET** tab or browse to find the C1.Phone.dll, C1.Phone.Chart.dll, and C1.Phone.Extended.dll files and click **OK**.
5. Select **View | Code** to open Code View. At the top of the file, add the following **Imports** statements (**using** in C#):

```
Imports C1.Phone
Imports C1.Phone.Chart
Imports C1.Phone.Extended
```

Adding XAML References

To add a XAML reference to the **ComponentOne Studio for Windows Phone** assemblies, complete the following steps:

1. Add the XAML namespace to the `<phone:PhoneApplicationPage>` tag by adding `xmlns:c1="clr-namespace:C1.Phone;assembly=C1.Phone"` so it appears similar to the following:

```
<phone:PhoneApplicationPage  x:Class="C1WP7.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
FontFamily="{StaticResource PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
Foreground="{StaticResource PhoneForegroundBrush}"
SupportedOrientations="Portrait" Orientation="Portrait"
shell:SystemTray.IsVisible="True">
```

Basic Windows Phone Application

When you initially create a new Windows Phone application, the XAML view for the application will appear similar to the following:

```
<phone:PhoneApplicationPage  x:Class="C1WP7.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
FontFamily="{StaticResource PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
```

```
Foreground="{StaticResource PhoneForegroundBrush}"
SupportedOrientations="Portrait" Orientation="Portrait"
shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page
title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0"
Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="MY APPLICATION"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="page name" Margin="9,-
7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1"
Margin="12,0,12,0"></Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <!--<phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton
IconUri="/Images/appbar_button1.png" Text="Button 1"/>
            <shell:ApplicationBarIconButton
IconUri="/Images/appbar_button2.png" Text="Button 2"/>
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="MenuItem 1"/>
                <shell:ApplicationBarMenuItem Text="MenuItem 2"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>-->

</phone:PhoneApplicationPage>
```

If you've worked in Silverlight before you'll note that it appears very similar to a standard Silverlight application. The **LayoutRoot** of the application is a Grid. By default when you create an application, it creates a **TitlePanel** and a **ContentPanel** containing panels and **TextBlock**s.

The markup that is commented out displays a default **ApplicationBar**. The **ApplicationBar** provides a menu system in Windows Phone. The Application Bar is displayed as a row of between one and four icon buttons along the bottom of the phone's screen. The icon buttons are used to provide users with quick access to an application's most common tasks.

Rotating Your Application

By default you application will not rotate when the Windows Phone is rotated. By default the **SupportedOrientations** property is set to **Portrait**, meaning the application is only displayed in portrait mode. Set the **SupportedOrientations** property to **PortraitOrLandscape** so that the application rotates when the phone rotates. For example, in XAML add `SupportedOrientations="PortraitOrLandscape"` to the `<phone:PhoneApplicationPage>` tag:

```
<phone:PhoneApplicationPage  x:Class="C1WP7.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
FontFamily="{StaticResource PhoneFontFamilyNormal}"
FontSize="{StaticResource PhoneFontSizeNormal}"
Foreground="{StaticResource PhoneForegroundBrush}"
SupportedOrientations="PortraitOrLandscape" Orientation="Portrait"
shell:SystemTray.IsVisible="True">
```

The Windows Phone has three orientation states: portrait and landscape (left or right). Your application may not appear as you would like it to when the phone is rotated. An easy solution for this is to create a templates to customize how your application will appear in portrait and landscape modes. You can us the **OrientationChanged** event to set the behavior of the application when the orientation of the Windows Phone changes from portrait to landscape and back.

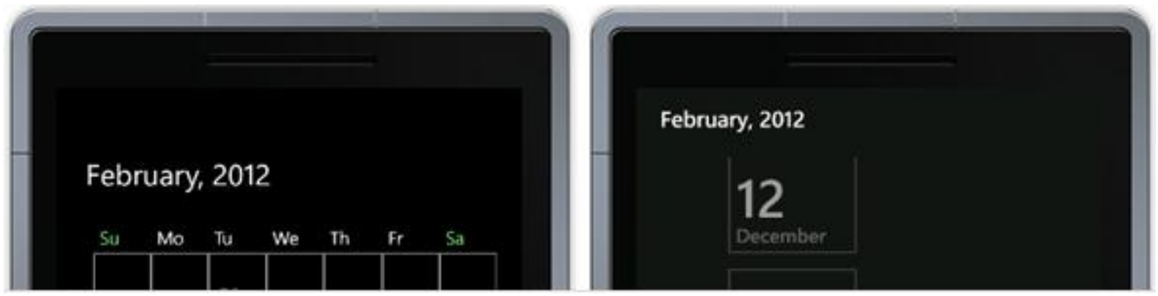
For more information and details, see the following Help Topic on MSDN: How to: Handle Orientation Changes on Windows Phone.

Key Features

Some of the main features of **ComponentOne Calendar for Windows Phone** that you may find useful include the following:

- **Gesture-based Month Navigation**

  By default, C1Calendar supports drag and flick gestures for date navigation. Tap the month header to display a date picker for quick month and year navigation.

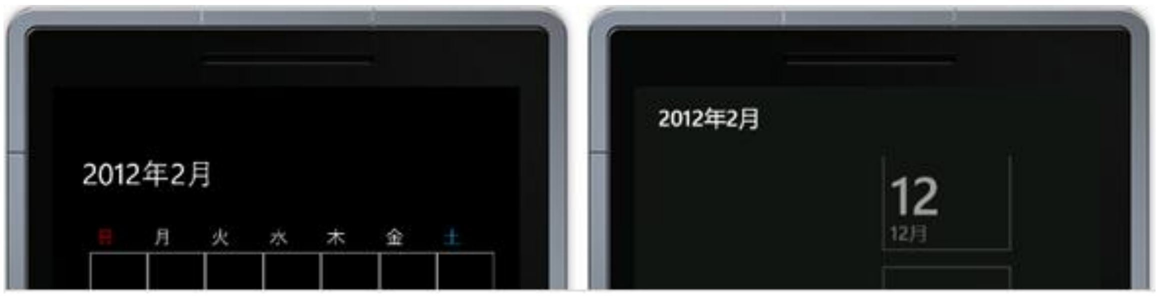


- **Date Range Selection**

  Enable the user to select one day or many days through a default tap or hold and slide gesture. Control the number of days they can select using the MaxSelectionCount property.

- **Customizable Calendar Settings**

  Specify the starting day of the week, the work weekdays collection, and more. You can use C1Calendar with any supported culture in .NET. Visually distinguish between work days and weekends through special brush properties. Make any date bolded to highlight it to the end-user.



- **Easy and Flexible Styling Model**

  With C1Calendar you can easily change control brushes without having to override templates. Each visible part of the control has its own brush, including adjacent days, weekends, selected days, the month header and more.

- **Customize Date Appearance and Content**

  You can alter the appearance of any individual day using a custom template and template selector. Use custom C1DataTemplateSelector implementations to display custom content within the date blocks, such as appointments from the phone's user data.

- **Show Week Numbers**

  Display week numbers by just setting one simple property. There are never any trailing empty weeks because C1Calendar always displays the minimum number of rows required per month whether it is 4, 5 or 6 weeks long.

Calendar for Windows Phone Quick Start

The following quick start guide is intended to get you up and running with **Calendar for Windows Phone**. In this quick start, you'll start in Visual Studio to create a new project with a C1Calendar control.

Step 1 of 3: Creating an Application with a C1Calendar Control

In this step, you'll create a Windows Phone application in Visual Studio using **Calendar for Windows Phone**.

Complete the following steps:

1. In Visual Studio 2010, select **File | New | Project** to open the **New Project** dialog box.
2. In the **New Project** dialog box, select a language in the left pane, and in the templates list select **Windows Phone Application**. Enter a **Name** for your project and click **OK**. The **New Windows Phone Application** dialog box will appear.
3. Click **OK** to close the **New Windows Phone Application** dialog box and create your project.
4. Right-click the project in the Solution Explorer and select **Add Reference**.
5. In the **Add Reference** dialog box, locate and select the **C1.Phone.dll** assembly and select **OK**.
6. Add the XAML namespace to the `<phone:PhoneApplicationPage>` tag by adding `xmlns:c1calendar="clr-namespace:C1.Phone.Calendar;assembly=C1.Phone.Calendar"` so it appears similar to the following:
```
   x:Class="CalendarWindowsPhone.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-
namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:c1calendar="clr-
namespace:C1.Phone.Calendar;assembly=C1.Phone.Calendar"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">
```
7. Edit the **TitlePanel** content to change the text in the **TextBlock** controls. It will appear similar to the following:
```
<!--TitlePanel contains the name of the application and page title-->
<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
```

```
    <TextBlock x:Name="ApplicationTitle" Text="ComponentOne Calendar for 
Windows Phone" Style="{StaticResource PhoneTextNormalStyle}"/>
    <TextBlock x:Name="PageTitle" Text="Calendar" Margin="9,-7,0,0" 
Style="{StaticResource PhoneTextTitle1Style}"/>
</StackPanel>
```

8. In the XAML window of the project, place the cursor between the `<Grid x:Name="ContentPanel"></Grid>` tags and click once.

9. Add the following XAML markup cursor between the `<Grid x:Name="ContentPanel"></Grid>` tags to add a C1Calendar control to the application:

   ```
   <Grid x:Name="ContentPanel">
           <c1calendar:C1Calendar x:Name="C1Calendar1" 
   Margin="12,0,12,0"></c1calendar:C1Calendar>
       </Grid>
   ```

   The calendar appears.

What You've Accomplished

You've successfully created a Windows Phone application containing a C1Calendar control. In the next step, you will Step 2 of 3: Adding Data to the Calendar, you will add the data for **C1Calendar**.

Step 2 of 3: Adding Data to the Calendar

In the last step, you added the **C1Chart** control to the application. In this step, you will add a **DataSeries** object and data for it.

To add data to the chart programmatically in the code behind file

1. Select **View | Code** to open the code editor.

2. Add the **C1.Phone.C1Chart** namespace directive

   ```
   using C1.Phone.Calendar
   ```

3. Add the following code under the namespace to create a Calendar chart:

   ```
   public partial class MainPage : PhoneApplicationPage
       {
           private static bool IsEmulator = 
   Microsoft.Devices.Environment.DeviceType == 
   Microsoft.Devices.DeviceType.Emulator;

           // dictionary of appointments for using in the Japanese calendar
           private Dictionary<DateTime, string> _boldedDays = null;

           // Constructor
           public MainPage()
           {
               InitializeComponent();
               if (IsEmulator)
               {
                   // add some bold days
                   cal1.BoldedDates.Add(DateTime.Today.AddDays(2));
                   cal1.BoldedDates.Add(DateTime.Today.AddDays(12));
                   cal1.BoldedDates.Add(DateTime.Today.AddDays(22));
                   cal1.BoldedDates.Add(DateTime.Today.AddDays(-2));
                   cal1.BoldedDates.Add(DateTime.Today.AddDays(-12));
                   cal1.BoldedDates.Add(DateTime.Today.AddDays(-22));
               }
           }
   ```

```
        private void SwitchToJapanese_Click(object sender, EventArgs e)
        {
            // change calendar culture
            Thread.CurrentThread.CurrentCulture = new
System.Globalization.CultureInfo("ja-JP");
            // show Sundays in Red and Saturdays in Blue
            DaySlotTemplateSelector datesSelector =
this.Resources["DaySlotTemplateSelector"] as DaySlotTemplateSelector;
            cal1.DaySlotTemplateSelector = datesSelector;
            // use bolded days dictionary defined in the
DaySlotTemplateSelector class instance
            this._boldedDays = datesSelector.BoldedDays;
            cal1.DayOfWeekSlotTemplateSelector = new
DayOfWeekTemplateSelector();
            cal1.WeekendBrush = new SolidColorBrush(Colors.Red);
            cal1.TodayBrush = new SolidColorBrush(Colors.Green);
            if (!IsEmulator)
            {
                // refresh bolded days
                if (_boldedDays != null)
                {
                    _boldedDays.Clear();
                }
                cal1.BoldedDates.BeginUpdate();
                cal1.BoldedDates.Clear();
                GetBoldedDates(cal1.DisplayDate.AddDays(-7),
cal1.DisplayDate.AddDays(49));
            }
        }

        private void Today_Click(object sender, EventArgs e)
        {
            cal1.Today();
        }

        private void cal1_Loaded(object sender, RoutedEventArgs e)
        {
            // update bolded dates for the visible month (and for adjacent
days)
            if (!IsEmulator)
            {
                cal1.BoldedDates.BeginUpdate(); // call BeginUpdate to
suspend refresh until getting all new dates
                GetBoldedDates(cal1.DisplayDate.AddDays(-7),
cal1.DisplayDate.AddDays(49));
            }
        }

        private void cal1_DisplayDateChanging(object sender,
DateChangedEventArgs e)
        {
            // update bolded dates for the new month (and for adjacent
month days)
            // (don't clear date in emulator)
            if (!IsEmulator)
            {
                if (_boldedDays != null)
                {
                    _boldedDays.Clear();
                }
                cal1.BoldedDates.BeginUpdate();
                cal1.BoldedDates.Clear();
                GetBoldedDates(e.NewValue.AddDays(-7),
e.NewValue.AddDays(49));
            }
        }

        private void cal1_DoubleTap(object sender, GestureEventArgs e)
        {
            // show the list of appointments for the bolded day
            FrameworkElement fel = e.OriginalSource as FrameworkElement;
            if (fel != null)
            {
                DaySlot slot = fel.DataContext as DaySlot;
                if (slot != null && slot.IsBolded)
                {
                    DateTime date = slot.Date;
                    Microsoft.Phone.UserData.Appointments appts = new
Microsoft.Phone.UserData.Appointments();
                    appts.SearchCompleted += (s, e1) =>
                    {
                        string message = date.ToShortDateString();
                        if (IsEmulator)
                        {
                            message += "\r\nNo information when run on
emulator";
                        }
                        else if
(e1.Results.Count<Microsoft.Phone.UserData.Appointment>() == 0)
                        {
                            message += "\r\nNo events for some reason";
                        }
                        else
                        {
                            foreach (Microsoft.Phone.UserData.Appointment
app in e1.Results)
                            {
                                message += "\r\n" + app.Subject;
                            }
                        }
                        MessageBox.Show(message);
                    };

                    appts.SearchAsync(date, date.AddDays(1).AddSeconds(-
1), null);
                }
            }
        }

        private void GetBoldedDates(DateTime start, DateTime end)
        {
            // get appointments from the device calendar
            Microsoft.Phone.UserData.Appointments appts = new
Microsoft.Phone.UserData.Appointments();

            appts.SearchCompleted += (s, e) =>
            {
                DateList boldedDates = cal1.BoldedDates;
                foreach (Microsoft.Phone.UserData.Appointment app in
e.Results)
                {
                    DateTime date = app.StartTime.Date;
                    if (_boldedDays != null)
                    {
                        // fill bolded days dictionary for Japanese
calendar
                        if (!_boldedDays.ContainsKey(date))
                        {
                            _boldedDays.Add(date, "");
                        }
                        string dayString = _boldedDays[date];
                        if (!String.IsNullOrEmpty(dayString) &&
!String.IsNullOrEmpty(app.Subject))
                        {
                            dayString = dayString + "\r\n";
                        }
                        _boldedDays[date] = dayString + app.Subject;
                    }
                    if (!boldedDates.Contains(date))
                    {
                        boldedDates.Add(date);
                    }
                }
                cal1.BoldedDates.EndUpdate(); // call EndUpdate to refresh
UI after changes
            };

            appts.SearchAsync(start, end, null);
        }

        private void Help_Click(object sender, EventArgs e)
        {
            MessageBox.Show("1. Tap to select single day.\r\n2. Tap or
hold and slide finger to select several days.\r\n3. Double tap on bolded
days to see details.");
        }
    }

    public class DaySlotTemplateSelector :
C1.Phone.Calendar.DaySlotTemplateSelector
    {
        public Dictionary<DateTime, string> BoldedDays = new
Dictionary<DateTime, string>();

        public override DataTemplate SelectTemplate(object item,
DependencyObject container)
        {
            DaySlot slot = item as DaySlot;
            if (slot != null && BoldedDays.ContainsKey(slot.Date))
            {
                // put appointments information into tag, so that it is
possible to show it in a day DataTemplate
                slot.Tag = BoldedDays[slot.Date];
            }
            else
            {
                // clear appointments information
                slot.Tag = null;
            }
            if (slot != null && !slot.IsAdjacent && slot.DayOfWeek ==
DayOfWeek.Saturday)
            {
                // set color for Saturday
                ((Control)container).Foreground = new
SolidColorBrush(Color.FromArgb(255, 0, 191, 255));
            }
            // the base class will select custom DataTemplate, defined in
the DaySlotTemplateSelector.Resources collection (see MainPage.xaml file)
            return base.SelectTemplate(item, container);
        }
    }

    public class DayOfWeekTemplateSelector :
C1.Phone.C1DataTemplateSelector
    {
        public override DataTemplate SelectTemplate(object item,
DependencyObject container)
        {
            DayOfWeekSlot slot = item as DayOfWeekSlot;
            if (slot != null && slot.DayOfWeek == DayOfWeek.Saturday)
            {
                // set color for Saturday
                ((Control)container).Foreground = new
SolidColorBrush(Color.FromArgb(255, 0, 191, 255));
            }
            // don't change DataTemplate at all
            return null;
        }
    }
}
```

In the next step, Step 3 of 3: Run the Application, you'll examine the Calendar for Windows Phone features.

What You've Accomplished

You have successfully added data to **C1Calendar**.

In the next step you'll observe some run-time behaviors.

Step 3 of 3: Run the Application

In this step you'll observe some run-time behaviors.

1. You can navigate through C1Calendar using drag or flick.
2. To select multiple days, tap or hold and then slide.

Calendar for Windows Phone Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio.NET environment.

Adding Bolded Dates to the C1Calendar

Use the C1Calendar.BoldedDates property to add bolded dates to the C1Calendar control like the following:

- Visual Basic

```
add some bold days
cal1.BoldedDates.Add(DateTime.Today.AddDays(2))
cal1.BoldedDates.Add(DateTime.Today.AddDays(12))
cal1.BoldedDates.Add(DateTime.Today.AddDays(22))
cal1.BoldedDates.Add(DateTime.Today.AddDays(-2))
cal1.BoldedDates.Add(DateTime.Today.AddDays(-12))
cal1.BoldedDates.Add(DateTime.Today.AddDays(-22))
```

- C#

```
// add some bold days
   cal1.BoldedDates.Add(DateTime.Today.AddDays(2));
   cal1.BoldedDates.Add(DateTime.Today.AddDays(12));
   cal1.BoldedDates.Add(DateTime.Today.AddDays(22));
   cal1.BoldedDates.Add(DateTime.Today.AddDays(-2));
   cal1.BoldedDates.Add(DateTime.Today.AddDays(-12));
   cal1.BoldedDates.Add(DateTime.Today.AddDays(-22));
```

Customizing Days Using DaySlotTemplateSelector

To custome the color of Sundays and today's date, complete the following:

- Visual Basic

```
show Sundays in Red and Today's date in Green
Dim datesSelector As DaySlotTemplateSelector =
TryCast(Me.Resources("DaySlotTemplateSelector"), DaySlotTemplateSelector)
cal1.DaySlotTemplateSelector = datesSelector
' use bolded days dictionary defined in the DaySlotTemplateSelector class
instance
Me._boldedDays = datesSelector.BoldedDays
cal1.DayOfWeekSlotTemplateSelector = New DayOfWeekTemplateSelector()
cal1.WeekendBrush = New SolidColorBrush(Colors.Red)
cal1.TodayBrush = New SolidColorBrush(Colors.Green)
```

- C#

```
// show Sundays in Red and Today's Date in Green
            DaySlotTemplateSelector datesSelector =
this.Resources["DaySlotTemplateSelector"] as DaySlotTemplateSelector;
            cal1.DaySlotTemplateSelector = datesSelector;
            // use bolded days dictionary defined in the
DaySlotTemplateSelector class instance
            this._boldedDays = datesSelector.BoldedDays;
            cal1.DayOfWeekSlotTemplateSelector = new
DayOfWeekTemplateSelector();
```

```
            cal1.WeekendBrush = new SolidColorBrush(Colors.Red);
            cal1.TodayBrush = new SolidColorBrush(Colors.Green);}
```