ComponentOne

# PdfViewer for Windows Phone

**ComponentOne, a division of GrapeCity**

201 South Highland Avenue, Third 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


Key Features ..... 1

PdfViewer for Windows Phone Quick Start ..... 1

- Step 1 of 3: Creating the C1PdfViewer Application ..... 1
- Step 2 of 3: Adding Content to the C1PdfViewer Control ..... 2
- Step 3 of 3: Running the C1PdfViewer Application ..... 3

Working with PdfViewer for Windows Phone ..... 5

- Touch Interaction ..... 5
- Basic Properties ..... 5
- Basic Events ..... 6
- Using PdfViewer in MVVM Applications ..... 6

PdfViewer Limitations ..... 6

PdfViewer for Windows Phone Layout and Appearance ..... 7

- Color Properties ..... 7
- Alignment Properties ..... 7
- Border Properties ..... 8
- Size Properties ..... 8

PdfViewer for Windows Phone Task-Based Help ..... 9

- Loading Documents into C1PdfViewer ..... 9
  - Loading Documents from Application Resources ..... 9
  - Loading Documents from Isolated Storage ..... 10
  - Loading Documents from the Web ..... 11
  - Loading PDF Files Created by C1PdfDocument ..... 12
- Finding Text in the Document ..... 12
- Using the Navigation Message Box Option ..... 16

# Key Features

**ComponentOne PdfViewer for Windows Phone** allows you to create customized, rich applications. Make the most of **PdfViewer for Windows Phone** by taking advantage of the following key features:

- **View and Save PDF Files**

  The **C1PdfViewer** control can be used to view and save PDF files on the Windows Phone device. **C1PdfViewer** has no external dependency on Adobe Reader to view or save files. Content is parsed and rendered as native XAML elements.

- **PDF Specification Support**

  **C1PdfViewer** supports a subset of the PDF 1.5 specification. There are a few important limitations including encryption, special fonts and rare image formats. Documents that use non-supported content will still render, but the formatting may be incorrect. It is recommended to use **C1PdfViewer** in a controlled environment where the features used by your PDF files can be tested before being used. The full list of limitations can be found in the documentation.

- **Find Text**

  Users can perform text searches within the document. As matches are found they are brought into view, and users can navigate through search results in a quick and intuitive manner.

- **Zoom Interaction**

  Users can double tap on the document to zoom in.

- **Get Pages from PDF**

  After loading a PDF, you can obtain a list of its pages as **FrameworkElements** to customize how the user views each page. Just call the **GetPages** method.

# PdfViewer for Windows Phone Quick Start

The following quick start guide is intended to get you up and running with **ComponentOne PdfViewer for Windows Phone**. In this quick start you'll create a simple project using a C1PdfViewer control. You'll create a new Windows Phone application, add the C1PdfViewer control to your application, add a PDF file that will be displayed in the C1PdfViewer control, and observe some of the run-time interactions possible with **PdfViewer for Windows Phone**.

## Step 1 of 3: Creating the C1PdfViewer Application

In this step you'll create a Windows Phone application using **PdfViewer for Windows Phone**. When you add a C1PdfViewer control to your application, you'll have a complete, functional document viewer interface that you can display PDF and HTML files in. To set up your project and add a C1PdfViewer control to your application, complete the following steps:

1. Create a new Windows Phone project in Visual Studio. In this example the application will be named "QuickStart". If you name the project something else, in later steps you may need to change references to "QuickStart" with the name of your project.

2. In the Solution Explorer, right-click the project name and choose **Add Reference**. In the **Add Reference** dialog box, locate and select the **C1.Phone.PdfViewer** assembly and click **OK** to add references to your project.
3. Open the XAML view of the  file; in this quick start you'll add the C1PdfViewer control using XAML markup.
4. Add the XAML namespace to the <phone:PhoneApplicationPage> tag with the following markup: `xmlns:c1pdfviewer="clr-namespace:C1.Phone.PdfViewer;assembly=C1.Phone.PdfViewer"`.

   The <phone:PhoneApplicationPage> tag will now appear similar to the following:

```
<phone:PhoneApplicationPage x:Class="QuickStart.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"
xmlns:c1pdfviewer="clr-
namespace:C1.Phone.PdfViewer;assembly=C1.Phone.PdfViewer" 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">
```

5. Add the `<c1:C1PdfViewer x:Name="C1PdfViewer1" />` tag within the Grid tags on the page to add the C1PDFViewer control to the application.

   The XAML will appear similar to the following:

```
<Grid>
    <c1pdfviewer:C1PdfViewer x:Name="C1PdfViewer1" />
</Grid>
```

   This will add a C1PdfViewer control named "C1PdfViewer1" to the application.

You've successfully set up your application's user interface, but if you run your application now you'll see that the C1PdfViewer control currently contains no content. In the next steps you'll add content to the C1PdfViewer control, and then you'll observe some of the run-time interactions possible with the control.

# Step 2 of 3: Adding Content to the C1PdfViewer Control

In the previous step you created a Windows Phone application and added the C1PdfViewer control to your project. In this step you'll add PDF content to the C1PdfViewer control. Note that in this step you will add a PDF file that is included with the **ComponentOne Studio for Windows Phone** samples, which are by default installed in the **Documents** or **MyDocuments** folder in the **ComponentOne Samples\Studio for \\PdfViewerSamples** directory. If you choose, you can instead use another PDF file and adapt the steps. To customize your project and add a PDF file to the C1PdfViewer control in your application, complete the following steps:

1. Navigate to the Solution Explorer, right-click the project name, and select **Add** | **Existing Item**.
2. In the **Add Existing Item** dialog box, locate the **C1XapOptimizer.pdf** file included in the **ControlExplorer** sample. In the file type drop-down box, you may need to choose **All Files** to view the PDF file. Note that if you choose, you can instead pick another PDF file to use.
3. In the Solution Explorer, click the PDF file you just added to the application. In the Properties window, set its **BuildAction** property to **Resource** and confirm that the **Copy to Output Directory** item is set to **Do not Copy**.
4. Switch to Code view by right-clicking the page and selecting **View Code**.

5. Add the following code to the main class :

   - Visual Basic

```
Public Sub New()
    Dim resource = Application.GetResourceStream(New
Uri("QuickStart;component/C1XapOptimizer.pdf", UriKind.Relative))
Me.C1PdfViewer1.LoadDocument(resource.Stream)
End Sub
```

   - C#

```
public MainPage()
{
    InitializeComponent();
    var resource = Application.GetResourceStream(new
Uri("QuickStart;component/C1XapOptimizer.pdf", UriKind.Relative));
    this.C1PdfViewer1.LoadDocument(resource.Stream);
}
```

   This code adds a stream and loads the stream into the C1PdfViewer control. Note that if you named the application differently, you will need to replace "QuickStart" with the name of your project. If you added a different PDF file, replace "C1XapOptimizer.pdf" with the name of your file.

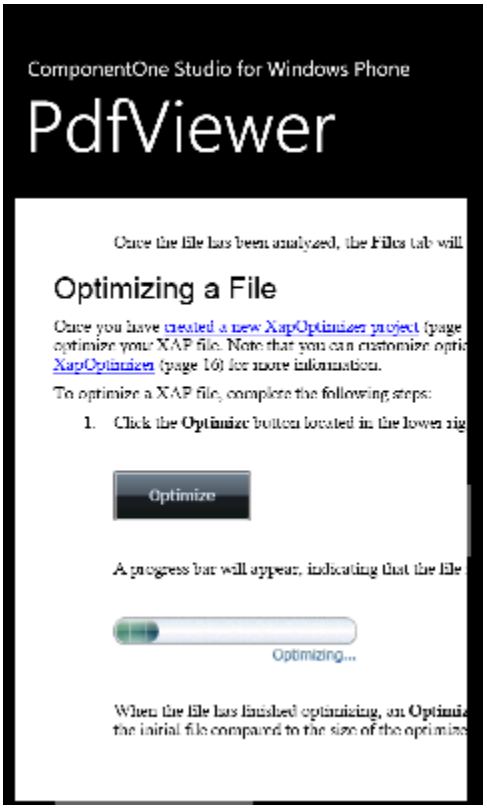
In this step you added content to the C1PdfViewer control. In the next step you'll view some of the run-time interactions possible in the control.

## Step 3 of 3: Running the C1PdfViewer Application

Now that you've created a Windows Phone application and added content to the C1PdfViewer control, the only thing left to do is run your application. To run your application and observe **PdfViewer for Windows Phone**'s run-time behavior, complete the following steps:

1. From the **Debug** menu, select **Start Debugging** to view how your application will appear at run time.

   The application will appear in the Windows Phone emulator similar to the following:




Notice that the PDF file you added appears in the content area of the control.

2. Tap and drag your finger (or click and drag the mouse in the emulator) to move the location of the PDF file.
3. Double-tap (or double-click in the emulator) on the PDF file to zoom in so that the PDF content is displayed larger.

Congratulations! You've completed the **PdfViewer for Windows Phone** quick start and created a simple Windows Phone application, added and customized a **PdfViewer for Windows Phone** control, and viewed some of the run-time capabilities of the control.

# Working with PdfViewer for Windows Phone

**ComponentOne PdfViewer for Windows Phone** includes the C1PdfViewer control, a simple viewer that allows you to load and view HTML and PDF files. When you add the C1PdfViewer control to a XAML window it exists as a fully functional input control that can be customized and include loaded content.

## Touch Interaction

**ComponentOne PdfViewer for Windows Phone** is optimized for a touch environment and includes several touch interactions. These interaction are fairly intuitive, for example to move to view a different area of a document you can touch and drag your finger across the screen of the phone and the section of the document that is displayed will change according to the direction you move.

You can also double-tap the screen to zoom into a document. The document will zoom in one level of magnification. To zoom in more, you can double-tap to zoom in another level of magnification. Additional interactions will be added in the future to improve the **PdfViewer for Windows Phone** touch environment.

## Basic Properties

**ComponentOne PdfViewer for Windows Phone** includes several properties that allow you to set the functionality of the control. Some of the more important properties are listed below. Note that you can see PdfViewer for Windows Phone Layout and Appearance (page 7) for more information about properties that control appearance.

The following properties let you customize the C1PdfViewer control:

| Property | Description |
|---|---|
| FindText | Gets or sets the text that will be searched by FindNext and FindPrevious methods. |
| HorizontalScrollBarVisibility | Gets or sets a value that indicates whether a horizontal ScrollBar should be displayed. |
| PageCount | Gets the current number of display pages for the content hosted by the C1PdfViewer. |
| PageMargin | Gets or sets the margin of the page used for displaying and printing HTML documents. |
| PageNumber | Gets the page number for the currently displayed page. |
| PageSeparation | Gets or sets the separation between pages. |
| PageSize | Gets or sets the size of the page used for displaying and printing HTML documents. |
| PageTemplate | Gets or sets the DataTemplate used to display pages. |
| SelectionBackground | Gets or sets a Brush for this C1PdfViewer control's selection. |
| VerticalScrollBarVisibility | Gets or sets a value that indicates whether a horizontal ScrollBar should be displayed. |
| ViewMode | Gets or sets the **ViewMode** for this C1PdfViewer. |

| | |
|---|---|
| ViewportHeight | Gets a value that contains the vertical size of the viewable content. |
| ViewportWidth | Gets a value that contains the horizontal size of the viewable content. |
| Zoom | Gets or sets the document zoom. |

## Basic Events

**ComponentOne PdfViewer for Windows Phone** includes events that allow you to set interaction and customize the control. Some of the more important events are listed below.

The following events let you customize the C1PdfViewer control:

| Event | Description |
|---|---|
| FindCountChanged | Event raised when the **FindCount** property has changed. |
| FindNumberChanged | Event raised when the **FindNumber** property has changed. |
| FindTextChanged | Event raised when the **FindText** property has changed. |
| IsFlowingChanged | Event raised when the **IsFlowing** property has changed. |
| PageCountChanged | Event raised when the **PageCount** property has changed. |
| PageNumberChanged | Event raised when the **PageNumber** property has changed. |
| RequestNavigate | Fired when a link from a document is clicked. |
| ViewModeChanged | Event raised when the **ViewMode** property has changed. |
| ZoomChanged | Event raised when the **Zoom** property has changed. |

## Using PdfViewer in MVVM Applications

You can use the **C1PdfViewer** control in MVVM applications by binding its Source property to some public property from your View Model. The Source should be bound to a property of type Uri.

For example, add a property to your View Model named "SourceUri" of type Uri:

```
public Uri SourceUri { get; set; }
```

In XAML, you would bind the C1PdfViewer to this property like this:

```
<c1:C1PdfViewer x:Name="pdfViewer" Source="{Binding SourceUri}" />
```

# PdfViewer Limitations

While **PDFViewer for Windows Phone** aims to provide a full-featured PDF viewer, it supports a subset of the PDF 1.5 standard and so, like most PDF viewers on the market, does have its limitations. These limitations focus in three areas: encryption, fonts, and images.

### Encrypted Files

Encrypted files are not compatible with **ComponentOne PdfViewer for Windows Phone**. At this time **C1PdfViewer** cannot open or save encrypted files.

### Fonts

**ComponentOne PdfViewer for Windows Phone** supports the following font types:

- **Embedded TrueType fonts:** These are fonts specified using the "FontFile2" mechanism in PDF.
- **Silverlight fonts:** This includes all font families supported by Silverlight.
- **PDF base fonts:** This includes fonts built into Adobe Acrobat such as Helvetica, Times, and Symbol.

The **C1PdfViewer** control does **not** support other font types available in the PDF specification, including Adobe Type 1 fonts (specified using the "FontFile" mechanism in the PDF file).

Documents that use non-supported fonts will still render, but the formatting will be incorrect (for example, the document may show overlapping text).

**Images**

**ComponentOne PdfViewer for Windows Phone** supports most common image types, including all binary stream formats supported by Silverlight as well as deflated streams of several types (RGB, Monochrome, and several common indexed formats).

The **C1PdfViewer** control does not support some rare formats such as deflated JPG streams, or advanced features such as custom color spaces or halftones. Note that scanned PDF files may contain TIFF data which the **C1PdfViewer** control is currently not capable of rendering.

# PdfViewer for Windows Phone Layout and Appearance

**ComponentOne PdfViewer for Windows Phone** includes several properties that allow you to customize the appearance of the control. You can change the color, border, and height of the control. The following topics describe some of these appearance properties.

## Color Properties

The following properties let you 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. |
| SelectionBackground | Gets or sets a Brush for this C1PdfViewer control's selection. |
| Foreground | Gets or sets a brush that describes the foreground color. This is a dependency property. |

## Alignment Properties

The following properties let you customize the control's alignment:

| Property | Description |
|---|---|
| HorizontalAlignment | Gets or sets the horizontal alignment characteristics applied to this element when it is composed within a parent element, such as a panel or items control. This is a dependency property. |
| VerticalAlignment | Gets or sets the vertical alignment characteristics applied to this element when it is composed within a parent element such as a panel or items control. |

| | |
|---|---|
| | 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 control:

| Property | Description |
|---|---|
| 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. |

# PdfViewer for Windows Phone Task-Based Help

The following task-based help topics assume that you are familiar with Visual Studio and Expression Blend and know how to use the C1PdfViewer control in general. If you are unfamiliar with the **ComponentOne PdfViewer for Windows Phone** product, please see the  first.

Each topic in this section provides a solution for specific tasks using the **ComponentOne PdfViewer for Windows Phone** product. Most task-based help topics also assume that you have created a new Windows Phone project and added a C1PdfViewer control to the project.

## Loading Documents into C1PdfViewer

If you run the application after adding a C1PdfViewer control to your application, you'll see an empty C1PdfViewer on the page. The next step is to invoke the LoadDocument method to add some content to the control. The LoadDocument method allows you to load content from **Stream** objects (which may contain PDF, HTML, or MHTML documents), or from **strings** (which may contain HTML or MHTML documents).

### Loading Documents from Application Resources

You can easily package an existing PDF file with your application and load it into **C1PdfViewer** at run time. For example, complete the following steps:

1. Navigate to the Solution Explorer, right-click the project name, and select **Add** | **New Folder**. Name the new folder "Resources".
1. Right-click the **Resources** folder in the Solution Explorer and select **Add** | **Existing Item**.
1. In the **Add Existing Item** dialog box, locate a PDF file. In the file type drop-down box, you may need to choose **All Files** to view the PDF file. Note that if you choose, you can instead pick another PDF file to use.
2. In the Solution Explorer, click the PDF file you just added to the application (in this example, we'll assume the file is named **MyPdf.pdf**). In the Properties window, set its **BuildAction** property to **Resource** and confirm that the **Copy to Output Directory** item is set to **Do not Copy**.
3. Switch to Code view by double-clicking on the preview in Design view.
4. Add the following imports statement to the top of the page:
    - Visual Basic
      ```
      Imports C1.Phone.PdfViewer
      ```
    - C#
      ```
      using C1.Phone.PdfViewer;
      ```
5. Add the following code to the main class:
    - Visual Basic
      ```
      Dim resource = Application.GetResourceStream(New
      Uri("/MyProject;component/Resources/MyPdf.pdf", UriKind.Relative))
      C1PdfViewer1.LoadDocument(resource.Stream)
      ```
    - C#

```
var resource = Application.GetResourceStream(new
Uri("/MyProject;component/Resources/MyPdf.pdf", UriKind.Relative));
c1PdfViewer1.LoadDocument(resource.Stream);
```

This code calls the **LoadDocument** method passing in the application resource stream.

**What You've Accomplished**

In this example you've loaded a PDF file into the **C1PdfViewer** from the application resources. You packaged an existing PDF file with your application in code and loaded it into the **C1PdfViewer** at run time.

## Loading Documents from Isolated Storage

You can save and load PDF files to isolated storage using the **C1PdfViewer** control. To save a file to isolated storage you would need to call the **SaveDocument** method passing in an isolated storage file stream:

- Visual Basic

```
Public Sub New()
    InitializeComponent()
    ' save file to isolated storage
    Using store = IsolatedStorageFile.GetUserStoreForApplication()
        Using stream = New IsolatedStorageFileStream("C1XapOptimizer.pdf",
FileMode.Create, FileAccess.Write, store)
            C1PdfViewer1.SaveDocument(stream)
        End Using
    End Using
End Sub
```

- C#

```
public MainPage()
{
    InitializeComponent();
    // save file to isolated storage
    using (var store = IsolatedStorageFile.GetUserStoreForApplication())
    {
        using (var stream = new
IsolatedStorageFileStream("C1XapOptimizer.pdf", FileMode.Create,
FileAccess.Write, store))
        {
            c1PdfViewer1.SaveDocument(stream);
        }
    }
}
```

To load an existing file from isolated storage call the **LoadDocument** method passing in an isolated storage file stream:

- Visual Basic

```
Public Sub New()
    InitializeComponent()
    '  load file from isolated storage
    Using store = IsolatedStorageFile.GetUserStoreForApplication()
          Using stream = New
IsolatedStorageFileStream("C1XapOptimizer.pdf", FileMode.OpenOrCreate,
FileAccess.Read, store)
            C1PdfViewer1.LoadDocument(stream)
          End Using
    End Using
End Sub
```

- C#

```
public MainPage()
{
    InitializeComponent();
    // load file from isolated storage
    using (var store = IsolatedStorageFile.GetUserStoreForApplication())
    {
        using (var stream = new
IsolatedStorageFileStream("C1XapOptimizer.pdf", FileMode.OpenOrCreate,
FileAccess.Read, store))
        {
            c1PdfViewer1.LoadDocument(stream);
        }
    }
}
```

**What You've Accomplished**

In this example you've saved and loaded PDF files to isolated storage using the **C1PdfViewer** control.

## Loading Documents from the Web

To load a file from the Web you must first download it to your application using an asynchronous request such as **WebClient** or **HttpWebRequest**. Then you simply pass the resulting stream to the **LoadDocument** method. The following code snippet example uses a **WebClient** type of request:

- Visual Basic

```
Public Sub New()
    InitializeComponent()
    ' load file from the Web
    Dim client As New WebClient()
    AddHandler client.OpenReadCompleted, AddressOf
client_OpenReadCompleted
    client.OpenReadAsync(New
Uri("http://www.componentone.com/newimages/Products/Documentation/Silverli
ght.Maps.pdf", UriKind.Absolute))
End Sub
Private Sub client_OpenReadCompleted(sender As Object, e As
OpenReadCompletedEventArgs)
      C1PdfViewer1.LoadDocument(e.Result)
End Sub
```

- C#

```
public MainPage()
{
    InitializeComponent();
    // load file from the Web
    WebClient client = new WebClient();
    client.OpenReadCompleted += new
OpenReadCompletedEventHandler(client_OpenReadCompleted);
    client.OpenReadAsync(new
Uri("http://www.componentone.com/newimages/Products/Documentation/Silverli
ght.Maps.pdf", UriKind.Absolute));
}
void client_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
    c1PdfViewer1.LoadDocument(e.Result);
}
```

**What You've Accomplished**

In this example you've loaded PDF files into the **C1PdfViewer** control from the Web.

### Loading PDF Files Created by C1PdfDocument

With the **ComponentOne PDF for Windows Phone** class library you can create PDF documents in code and save them out to a stream. The following code snippet shows how to create a simple document and load it into **C1PdfViewer** without having to save it into isolated storage or put on the Web:

- Visual Basic

```
Public Sub New()
    InitializeComponent()
    ' create new C1PdfDocument
    Dim doc As New C1PdfDocument()
    ' add some content to PDF
    doc.DrawString("Hello World!", New Font("Arial", 14), Colors.Black,
doc.PageRectangle)
    ' save PDF to memory stream
    Dim ms As New MemoryStream()
doc.Save(ms)
    ' load PDF from stream
    ms.Seek(0, SeekOrigin.Begin)
    c1PdfViewer1.LoadDocument(ms)
End Sub
```

- C#

```
public MainPage()
{
    InitializeComponent();
    // create new C1PdfDocument
    C1PdfDocument doc = new C1PdfDocument();
    // add some content to PDF
    doc.DrawString("Hello World!", new Font("Arial", 14), Colors.Black,
doc.PageRectangle);
    // save PDF to memory stream
    MemoryStream ms = new MemoryStream();
doc.Save(ms);
    // load PDF from stream
    ms.Seek(0, SeekOrigin.Begin);
c1PdfViewer1.LoadDocument(ms);
}
```

**What You've Accomplished**

In this example you've loaded PDF files into the **C1PdfViewer** control rom the Web.

# Finding Text in the Document

You can easily allow end-users to search for specific text or characters in PDF documents. This topic will walk you through adding the XAML markup and the code to create a search function.

Note that in this step you will add a PDF file that is included with the **ComponentOne Studio for Windows Phone** samples, which are by default installed in the **Documents** or **MyDocuments** folder in the **ComponentOne Samples\Studio for Windows Phone\General\CS\ControlExplorer\Resources** directory. If you choose, you can instead use another PDF file and adapt the steps.

To customize your project and add a PDF file to the C1PdfViewer control in your application, complete the following steps:

1. In the Solution Explorer, right-click on the project name and select **Add | Existing Item** from the menu. The **Add Existing Item** dialog box will appear.
2. Select the sample PDF file, **C1XapOptimizer.pdf**, and click **Add**.
3. Right-click the PDF file you just added in the Solution Explorer and select **Properties** from the menu. Set the **BuildAction** property to **Resource** and confirm that the **Copy to Output Directory** item is set to **Do Not Copy**.
4. Add the following markup after the <Grid.RowDefinitions> tags to create the grid structure for the search function and to add a C1PdfViewer to the page.

```
<Grid Grid.Row="0" x:Name="panelFindText" >
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="Auto" />
    </Grid.ColumnDefinitions>
    <Button x:Name="btnClose" Content="x" Click="btnClose_Click"
BorderThickness="0" />
    <TextBox x:Name="txtSearch" TextChanged="txtSearch_TextChanged"
Grid.Column="1"  />
    <Button Content="next" Click="btnDoSearch_Click"
x:Name="btnDoSearch" Grid.Column="2" />
</Grid>
<Grid Grid.Row="1">
    <TextBlock x:Name="txtLoading" Text="Loading..."
VerticalAlignment="Center" HorizontalAlignment="Center" />
    <c1pdfviewer:C1PdfViewer x:Name="pdfViewer" />
    <Border VerticalAlignment="Top" HorizontalAlignment="Left"
Background="{StaticResource PhoneChromeBrush}" CornerRadius="5"
Opacity="0.8" Margin="5" Grid.Row="1" >
            <StackPanel Orientation="Horizontal" Margin="4" >
                <TextBlock Text="{Binding ElementName=pdfViewer,
Path=PageNumber}" Foreground="{StaticResource PhoneForegroundBrush}" />
                <TextBlock Text=" / " Foreground="{StaticResource
PhoneForegroundBrush}" />
                <TextBlock Text="{Binding ElementName=pdfViewer,
Path=PageCount}" Foreground="{StaticResource PhoneForegroundBrush}" />
            </StackPanel>
        </Border>
    </Grid>
</Grid>
```

5. Switch to Code View by right-clicking the page and selecting **View Code** from the menu.
6. Add the following imports statement to the top of the page:
   - Visual Basic
     ```
     Imports C1.Phone.PdfViewer
     ```
   - C#
     ```
     using C1.Phone.PdfViewer;
     ```
7. Add the following code to the main class:
   - Visual Basic
     ```
     Dim btnShowFindText As New ApplicationBarMenuItem("find text")
     ```
   - C#

```
ApplicationBarMenuItem btnShowFindText = new
ApplicationBarMenuItem("find text");
```

8. Add the following code beneath the **InitializeComponent()** method:

- Visual Basic

```
btnShowFindText.Click += new EventHandler(this.btnShowFindText_Click);
 btnClose_Click(null, new RoutedEventArgs());
  PhoneApp2_Loaded(object, sender, RoutedEventArgs, e);
 {var;
 resource = Application.GetResourceStream(new
Uri("PhoneApp2;component/C1XapOptimizer.pdf", UriKind.Relative));
 pdfViewer.LoadDocument(resource.Stream);
 //pdfViewer.PageRendered = (pdfViewer.PageRendered + s2);
 ,e2;
 Unknown=Greater{//if ((e2.TotalRenderedPages >= 2)) {
     txtLoading.Visibility = System.Windows.Visibility.Visible;
     pdfViewer.Visibility = System.Windows.Visibility.Visible;
     void;
     txtSearch_TextChanged(object, sender, TextChangedEventArgs, e);
     btnDoSearch.IsEnabled = (txtSearch.Text.Trim().Length > 0);
     void;
     btnDoSearch_Click(object, sender, RoutedEventArgs, e);
     pdfViewer.FindText = txtSearch.Text;
     pdfViewer.FindNextCommand.Execute(null);
     void;
     btnClose_Click(object, sender, RoutedEventArgs, e);
     panelFindText.Visibility = System.Windows.Visibility.Visible;
     btnShowFindText.IsEnabled = true;
     void;
     btnShowFindText_Click(object, sender, EventArgs, e);
     panelFindText.Visibility = System.Windows.Visibility.Visible;
     btnShowFindText.IsEnabled = false;
     // TODO: #region ... Warning!!! not translated
     IExposeApplicationBarItems;
     Members;
     (IEnumerable
                 < (IApplicationBarMenuItem > ApplicationBarItems));
     yield;
     return btnShowFindText;
 }
```

- C#

```
 btnShowFindText.Click += new EventHandler(btnShowFindText_Click);
                btnClose_Click(null, new RoutedEventArgs());
void PhoneApp2_Loaded(object sender, RoutedEventArgs e)
        {
 var resource = Application.GetResourceStream(new
Uri("PhoneApp2;component/C1XapOptimizer.pdf", UriKind.Relative));
            pdfViewer.LoadDocument(resource.Stream);
            //pdfViewer.PageRendered += (s2, e2) =>
            {
                //if (e2.TotalRenderedPages >= 2)
                {
                    txtLoading.Visibility =
System.Windows.Visibility.Visible;
```

```
                    pdfViewer.Visibility =
System.Windows.Visibility.Visible;
                }
            };
        }
          private void txtSearch_TextChanged(object sender,
TextChangedEventArgs e)
        {
            btnDoSearch.IsEnabled = (txtSearch.Text.Trim().Length > 0);
        }
        private void btnDoSearch_Click(object sender, RoutedEventArgs
e)
        {
            pdfViewer.FindText = txtSearch.Text;
            pdfViewer.FindNextCommand.Execute(null);
        }
        private void btnClose_Click(object sender, RoutedEventArgs e)
{
            panelFindText.Visibility =
System.Windows.Visibility.Visible;
            btnShowFindText.IsEnabled = true;
        }
        void btnShowFindText_Click(object sender, EventArgs e)
        {
            panelFindText.Visibility =
System.Windows.Visibility.Visible;
            btnShowFindText.IsEnabled = false;
        }
        #region IExposeApplicationBarItems Members
        public IEnumerable<IApplicationBarMenuItem> ApplicationBarItems
        {
            get
            {
                yield return btnShowFindText;
            }
        }
    }
}
```

9. Press F5 to run your application. The application should resemble the following image:


Note that you can search for specific text or characters in the document. When it is found, it will be highlighted.

# Using the Navigation Message Box Option

**C1PdfViewer** allows you to add a message box to control navigation away from the PDF document. If a user clicks on a link in the PDF file, the application will give them the option to navigate away from the document or to cancel the navigation request.

Note that in this step you will add a PDF file that is included with the **ComponentOne Studio for Windows Phone** samples, which are by default installed in the **Documents** or **MyDocuments** folder in the **ComponentOne Samples\Studio for Windows Phone \ \PdfViewerSamples** directory. If you choose, you can instead use another PDF file and adapt the steps.

Complete these steps to create the navigation request:

1. In the Solution Explorer, right-click on the project name and select **Add | Existing Item** from the menu. The **Add Existing Item** dialog box will appear.
2. Select the sample PDF file, **C1XapOptimizer.pdf**, and click **Add**.

3. Right-click the PDF file you just added in the Solution Explorer and select **Properties** from the menu. Set the **BuildAction** property to **Resource** and confirm that the **Copy to Output Directory** item is set to **Do Not Copy**.
4. Open the XAML view of the **Mainpage.xaml** file and add the following markup to create the C1PdfViewer component:

```
<c1pdfviewer:C1PdfViewer x:Name="pdfViewer" ViewMode="FitWidth" />
```

5. Open the Code View by right-clicking on the document and add the following imports statement:

- Visual Basic

```
Imports C1.Phone.PdfViewer
```

- C#

```
using C1.Phone.PdfViewer;
```

6. Add the following code after the **InitializeComponent()** method:

- Visual Basic

```
Loaded = (Loaded + AddressOf Me.PhoneApp5_Loaded)
 pdfViewer.RequestNavigate = (pdfViewer.RequestNavigate + s)
 ,e
 Unknown=Greater{Dim result As var = MessageBox.Show(("The document is
requesting to navigate to " + e.Uri), "Navigate",
MessageBoxButton.OKCancel)
 If (result = MessageBoxResult.OK) Then
     ' navigate to urls loading web browser control
     Dim browser As WebBrowser = New WebBrowser
     browser.Navigate(e.Uri)
     LayoutRoot.Children.Clear
     LayoutRoot.Children.Add(browser)
 End If
 UnknownUnknownUnknownUnknown

     Private Sub PhoneApp5_Loaded(ByVal sender As Object, ByVal e As
RoutedEventArgs)
         Dim resource As var = Application.GetResourceStream(New
Uri("PhoneApp5;component/C1XapOptimizer.pdf", UriKind.Relative))
         pdfViewer.LoadDocument(resource.Stream)
         'pdfViewer.PageRendered += (s2, e2) =>
         'if (e2.TotalRenderedPages >= 2)
         pdfViewer.Visibility = System.Windows.Visibility.Visible
     End Sub
```

- C#

```
Loaded += new RoutedEventHandler(PhoneApp5_Loaded);

                pdfViewer.RequestNavigate += (s, e) =>
                {
                    var result = MessageBox.Show("The document is
requesting to navigate to " + e.Uri, "Navigate",
MessageBoxButton.OKCancel);
                    if (result == MessageBoxResult.OK)
                    {
                        // navigate to urls loading web browser control
                        WebBrowser browser = new WebBrowser();
                        browser.Navigate(e.Uri);

                        LayoutRoot.Children.Clear();
                        LayoutRoot.Children.Add(browser);
```

```
                    }
                };
            }
                void PhoneApp5_Loaded(object sender, RoutedEventArgs e)
            {
    var resource = Application.GetResourceStream(new
Uri("PhoneApp5;component/C1XapOptimizer.pdf", UriKind.Relative));
                pdfViewer.LoadDocument(resource.Stream);
                //pdfViewer.PageRendered += (s2, e2) =>
                {
                    //if (e2.TotalRenderedPages >= 2)
                    {
                       pdfViewer.Visibility =
System.Windows.Visibility.Visible;

        }
    }
}
    }
}
```

7. Press F5 to run your application. Locate a link in the PDF document and click on it. The Navigation Message Box should appear as in the following image:


Navigate
The document is requesting to navigate to http://c1verification.componentone.com/webfiles/Activate.aspx
ok
cancel
Internet
From our web site
By e-mail
Over the phone