ComponentOne

FilePicker for WPF

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

ComponentOne FilePicker for WPF Overview

Enable file selection in your desktop apps with **ComponentOne FilePicker™ for WPF**. The **C1FilePicker** control is similar to a combo box, except instead of showing a drop-down list, it shows a file picker dialog box.

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

Getting Started

Get started with the following topics:

- Key Features (page 1)
- Quick Start (page 3)
- Task-Based Help (page 16)

Help with ComponentOne Studio for WPF

Getting Started

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

What's New

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

Key Features

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

- **Filter File Types**

  Limit the file types you search for based on a file extension or category. For more information, see File Filtering (page 11) and for an example of adding a filter for image files, see the Adding a File Filter (page 18) topic.

- **Multi-file Selection**

  Allow end-users to select a single file at a time or multiple files at once to speed up the process. For more information, see Multiple File Selection (page 9) and for an example of allowing users to select multiple files in the C1FilePicker control, see the Selecting Multiple Files (page 17) topic.

- **Watermark Support**

  Specify a watermark to help aid the user in selecting a file. For more information, see Watermark Text (page 9) and for an example of removing the watermark, see the Removing the Watermark (page 16) topic.

- **Easily Change Colors with ClearStyle**

  Allow end-users to select a single file at a time or multiple files at once to speed up the process. For more information, see ComponentOne ClearStyle Technology (page 14).

FilePicker for WPF Quick Start

The following quick start guide is intended to get you up and running with **ComponentOne FilePicker for WPF**. In this quick start you'll create an application that allows you to select and view thumbnails of images on your computer. You'll create a new project in Visual Studio, add and customize the **C1FilePicker** control, and add controls to view files selected with the **C1FilePicker** control.

Step 1 of 3: Creating a WPF Application

In this step you'll begin in Visual Studio to create a WPF application which will use **ComponentOne FilePicker for WPF** to select image files on your computer to view. You'll create a new WPF project and controls to your application.

To set up and add controls to your application, complete the following steps:

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select a language in the left pane, and in the templates list select **WPF Application**. Enter a **Name** for your project, for example "QuickStart", and click **OK**. The project will be created and the **MainWindow.xaml** file should open.
3. In the XAML window of the project, place the cursor between the `<Grid>` and `</Grid>` tags and click once.
4. Add the following XAML markup between the `<Grid>` and `</Grid>` tags in the **MainWindow.xaml** file:

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition />
    <RowDefinition Height="Auto" />
</Grid.RowDefinitions>
```

   This markup creates row definitions in the grid to keep controls in separate areas.
5. Navigate to the Toolbox and double-click the **C1FilePicker** icon to add the control to your application.
6. Edit the **C1FilePicker** tag so it appears similar to the following:

```
<c1:C1FilePicker x:Name="C1FilePicker1" Grid.Row="0" Margin="15"
Height="30" SelectedFilesChanged="C1FilePicker_SelectedFilesChanged" />
```

   The markup above gives the control a name, specifies the grid row the control appears in, sets the height of the control and the margin around the control, and indicates an event handler for the SelectedFilesChanged event. You'll add the event handler code in a later step.
7. Add the following XAML markup beneath the `<c1:C1FilePicker>` tag and before the `</Grid>` tag in the **MainPage.xaml** file:

```
<ScrollViewer Grid.Row="1" Margin="15,0,15,0">
    <ListBox x:Name="ListBox" />
</ScrollViewer>
<StackPanel Grid.Row="2" Name="stackPanel1" Orientation="Horizontal"
HorizontalAlignment="Center">
    <Button Content="Clear File Selection" Height="30"
Margin="0,15,15,15" Name="button1" Width="150" Grid.Row="2"
Click="button1_Click" />
    <Button Content="Clear List Items" Height="30" Margin="15,15,0,15"
Name="button2" Width="150" Grid.Row="2" Click="button2_Click" />
</StackPanel>
```

This markup will add a **ListBox** control which will allow you to view local images you select using the **C1FilePicker** control. The markup also adds two buttons which will let you clear the content of the **C1FilePicker** control and the content of the **ListBox**.

What You've Accomplished

You've successfully created and set up a WPF application and added controls to the page. In the next step you'll add code to add functionality to your application.

Step 2 of 3: Adding Code to the Application

In the last step you set up a WPF application, but if you run your application the controls currently do nothing. In this step you'll continue by adding code to add functionality to the application.

Complete the following steps:

1. Navigate to the Solution Explorer, right-click **MainWindow.xaml** file, and select **View Code** to switch to Code view.
2. In Code view, add the following import statements to the top of the page if they are not included:
   - Visual Basic
     ```
     Imports System.Windows.Media.Imaging
     Imports C1.WPF
     ```
   - C#
     ```
     using System.Windows.Media.Imaging;
     using C1.WPF;
     ```
3. Add the following event handler to the **MainWindow.xaml.cs** (or **.vb**) file, below all the other methods in the **MainPage** class:
   - Visual Basic
     ```
     Private Sub C1FilePicker_SelectedFilesChanged(sender As Object, e As
     EventArgs)
         If C1FilePicker1.SelectedFile IsNot Nothing Then
             Dim stream = C1FilePicker1.SelectedFile.OpenRead()
             Dim source = New BitmapImage()
             source.BeginInit()
             source.StreamSource = stream
             source.EndInit()
             Dim image = New Image()
             image.Source = source
             image.Stretch = Stretch.Uniform
             image.Height = 100
             ListBox.Items.Add(image)
         End If
     End Sub
     ```
   - C#
     ```
     private void C1FilePicker_SelectedFilesChanged(object sender, EventArgs
     e)
     {
         if (C1FilePicker1.SelectedFile != null)
         {
             var stream = C1FilePicker1.SelectedFile.OpenRead();
             var source = new BitmapImage();
             source.BeginInit();
             source.StreamSource = stream;
             source.EndInit();
     ```

```
        var image = new Image();
        image.Source = source;
        image.Stretch = Stretch.Uniform;
        image.Height = 100;
        ListBox.Items.Add(image);
    }
}
```

This code handles the SelectedFilesChanged event. When a user selects an image with the **C1FilePicker** control, the image is customized and added to the **ListBox** control.

4. Add the following code to handle the **Click** events of the **Button** controls on the page:

- Visual Basic

```
Private Sub button1_Click(sender As Object, e As RoutedEventArgs)
    C1FilePicker1.ClearSelection()
End Sub

Private Sub button2_Click(sender As Object, e As RoutedEventArgs)
    ListBox.Items.Clear()
End Sub
```

- C#

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    C1FilePicker1.ClearSelection();
}

private void button2_Click(object sender, RoutedEventArgs e)
{
    ListBox.Items.Clear();
}
```

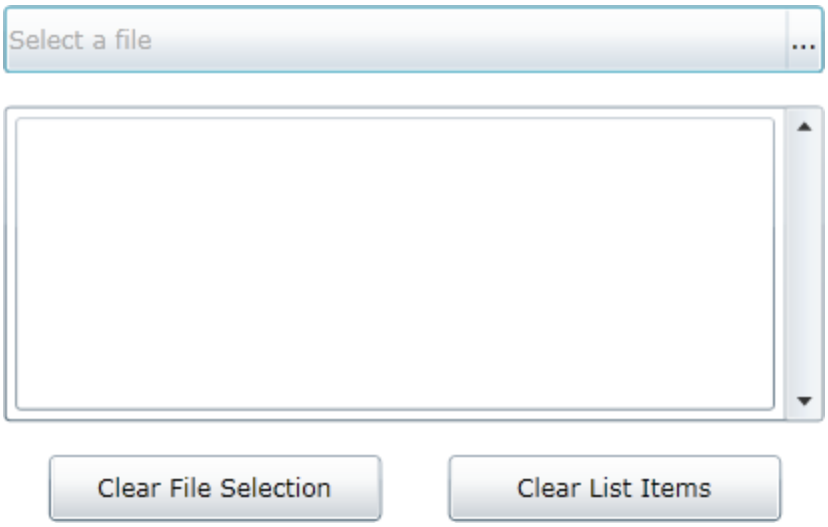
What You've Accomplished

In this step you added code to add functionality to your application. In the next step you'll run your application and observe some of the run-time interactions possible with **ComponentOne FilePicker for WPF**.

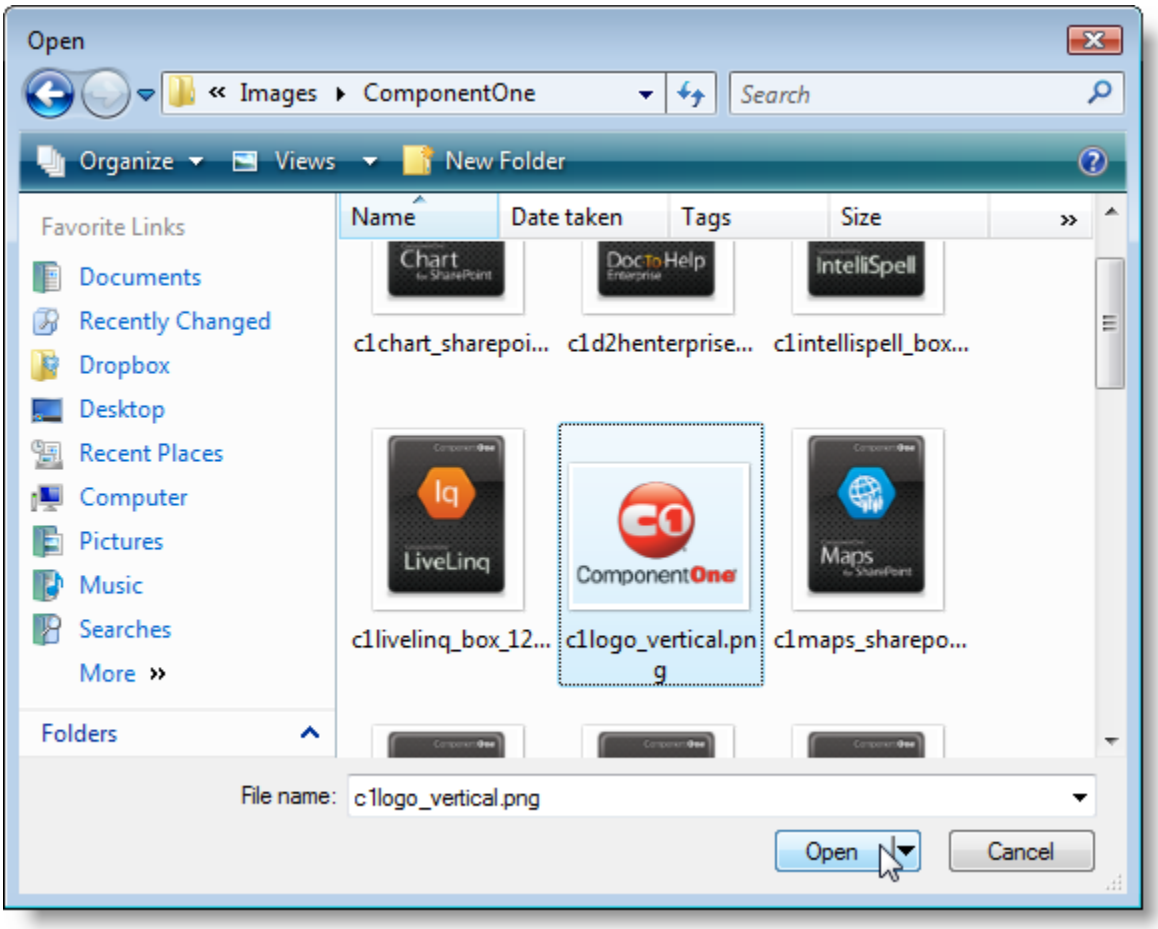
Step 3 of 3: Running the Application

Now that you've created a WPF application and, set up the application, and added code to add functionality to the application, the only thing left to do is run your application. To observe your application's run-time interactions, complete the following steps:

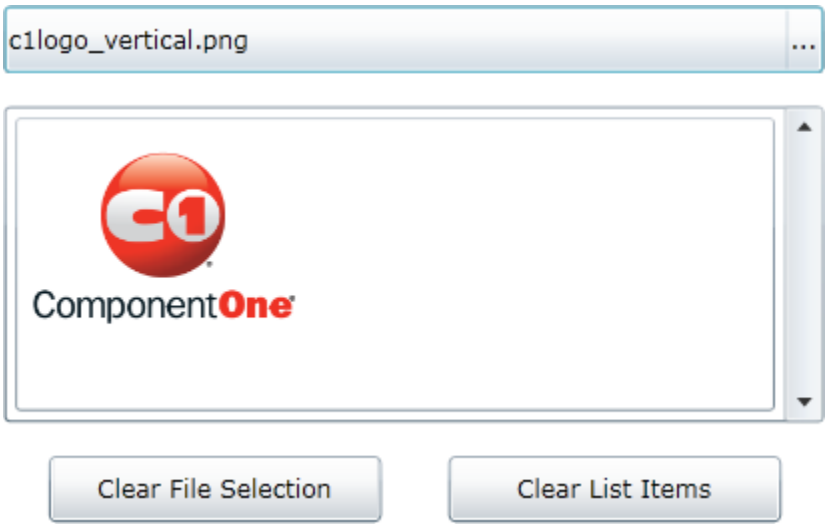
1. Choose **Debug | Start Debugging** from the menu to run your application. The application will appear similar to the following image:



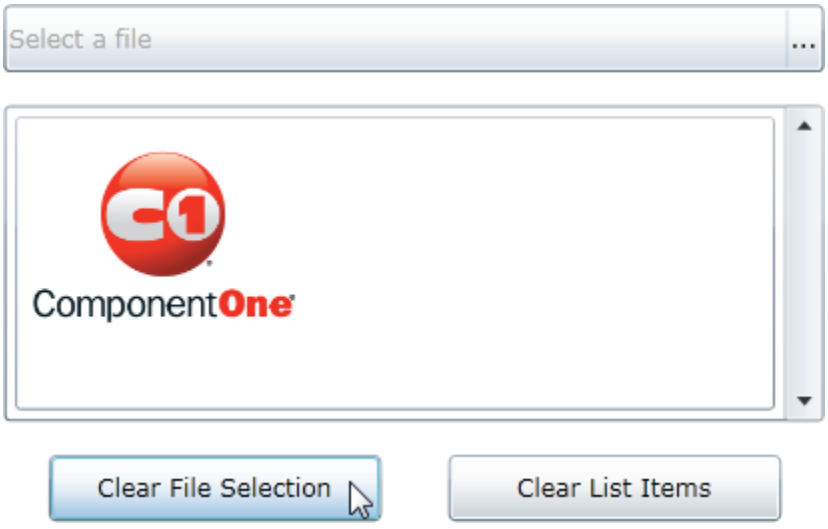
2. Click the ellipses button (**…**) on the **C1FilePicker** control. The **Open** dialog box will appear.
3. In the **Open** dialog box, locate and select an image, and click **Open** to open the selected image:



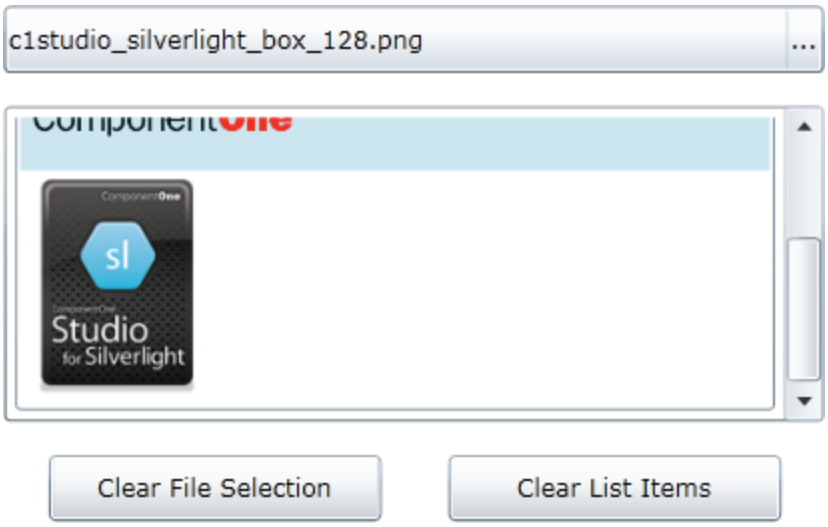
4. The selected image will appear in the list box and the name of the file will appear in the **C1FilePicker** control:



5. Click the **Clear File Selection** button. Notice that the name of the file no longer appears in the **C1FilePicker** control:



6. Click the ellipses button on the **C1FilePicker** control again. The **Open** dialog box will appear.
7. In the **Open** dialog box, try to select more than one image. Notice that you cannot. That is because the Multiselect property is sent to **False** by default. To select multiple files at one, you would need to set the Multiselect property to **True**.
8. Select a file in the **Open** dialog box and click **Open**. Notice that the second file appears listed in the **C1FilePicker** control and has been added to the List Box:



9. Click the **Clear List Items** button; the list of files will be cleared.

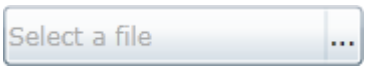
What You've Accomplished

Congratulations, you've completed the **FilePicker for WPF** quick start! You've created a simple application that uses **FilePicker for WPF** to select image files that are then displayed in another control.

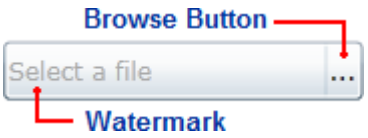
To learn more about the features and functionality of **ComponentOne FilePicker for WPF**, see the Working with FilePicker for WPF (page 11) topic. For examples of specific customizations, see the FilePicker for WPF Task-Based Help (page 16) topic.

Elements and Selection

ComponentOne FilePicker for WPF provides the ability to select files. Selected files can then be uploaded with **ComponentOne Upload for WPF** or used by other controls. The basic C1FilePicker control appears similar to the following image:

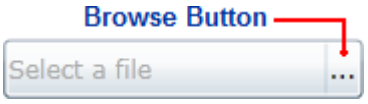


The control consists of a box that lists a watermark or the selected file name(s) and a **Browse** button indicated by an **ellipsis**:



Browse Button

The **Browse** button appears to the right of the C1FilePicker control and appears as an **ellipses** button:

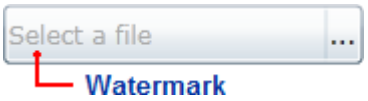


When a user clicks the **Browse** button at run time, the **OpenFileDialog** dialog box will appear. The **OpenFileDialog** allows users to choose one or more files on the local computer or a networked computer that will be added to the SelectedFiles collection. For more information, see the Open File Dialog Box (page 10) topic.

You can customize the content and appearance of the **Browse** button by setting the BrowseContent property to an object. The object you select will appear as the browse button.

Watermark Text

The watermark appears in the text area of the C1FilePicker control by default. The watermark can give directions or suggestions for users at run time, for example the default watermark text states "Select a file":



Notice that the text appears grayed out. When a file or multiple files are selected the file(s) will appear in this text area instead and appear a darker color to be distinguished from the watermark. If a file is selected and the files are cleared from the C1FilePicker control using the ClearSelection method, the watermark will appear again. See Clearing Selected Files (page 17) for an example.

You can customize the watermark text by setting the Watermark property to the value you want to appear. For an example, see the Removing the Watermark (page 16) topic. If you do not want a watermark to appear, you can set the Watermark property to a blank value (for example, a space character).

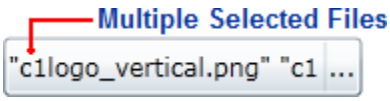
Selected Files

When a file or multiple files are selected the file(s) will appear in this text area instead and appear a darker color to be distinguished from the watermark. For example, when a file is chosen the C1FilePicker control appears similar to the following image:

Note that the name of the selected file appears in the text area of the C1FilePicker control. To get the name of the selected file, you can use the SelectedFile property. To clear selected files, you can use the ClearSelection method. When the selected files are cleared, the watermark will appear displayed in the C1FilePicker control by default. See Clearing Selected Files (page 17) for an example.

Multiple File Selection

When the Multiselect property is set to **True**, the user can choose multiple files in the **OpenFileDialog** dialog box at run time. When multiple files are selected, they all appear in the text area of the C1FilePicker control, separated by commas, and in quotation marks. For example, two files are selected in the following image:

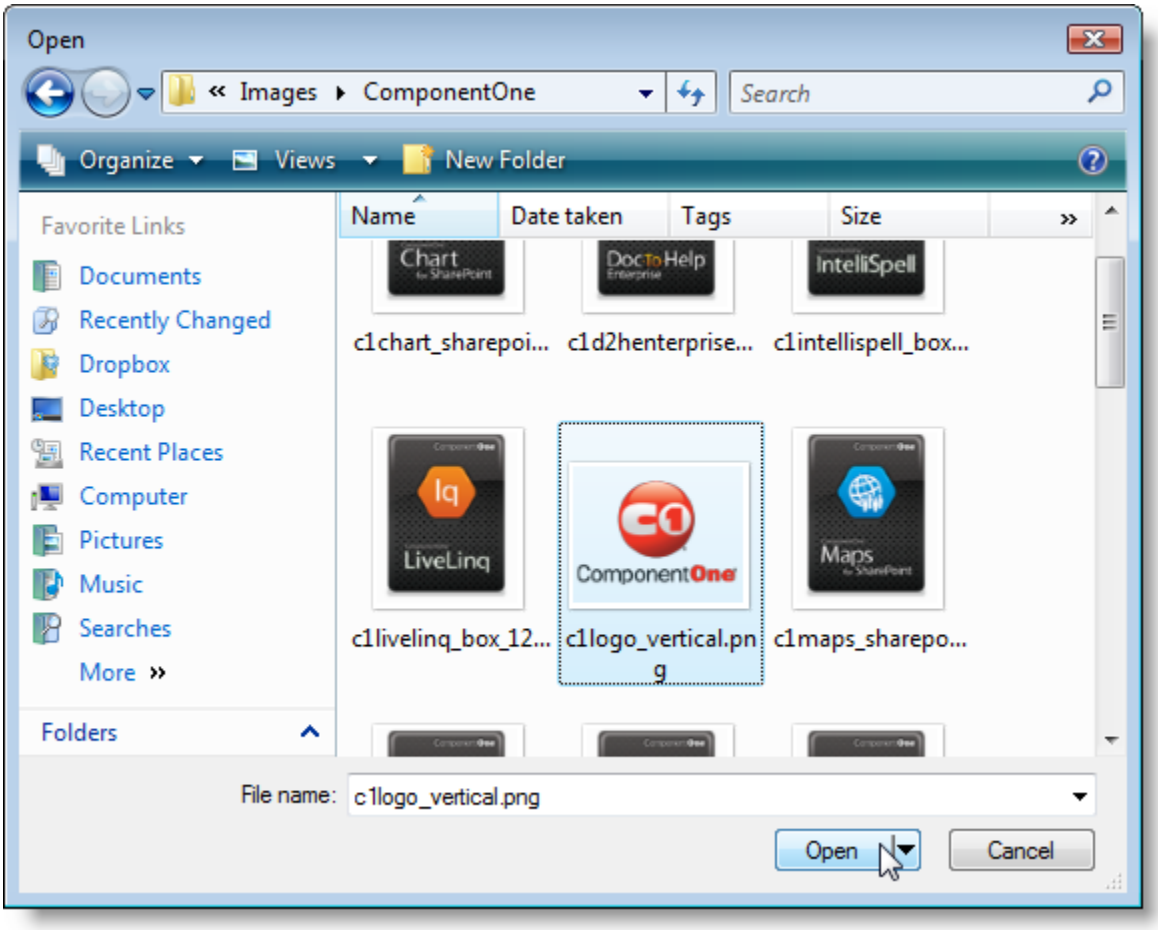


For an example of selecting multiple files, see the Selecting Multiple Files (page 17) topic. To clear selected files, you can use the ClearSelection method. When the selected files are cleared, the watermark will appear displayed in the C1FilePicker control by default. See Clearing Selected Files (page 17) for an example.

Open File Dialog Box

ComponentOne FilePicker for WPF uses the standard Microsoft **OpenFileDialog** box to enable to users to select files. The **OpenFileDialog** dialog box enables users to open one or more files on a the local computer or a networked computer. Files selected in the **OpenFileDialog** dialog box are then added to the SelectedFiles collection.

The **OpenFileDialog** dialog box appears similar to a traditional dialog box. For example, the dialog box appears similar to the following image:



You can choose to add a files by navigating to the folder where they are located, selecting the files to add, and clicking the **Open** button. To select multiple files in the **OpenFileDialog** dialog box, you must sent the Multiselect property to **True**.

You can apply a filter in the **OpenFileDialog** dialog box so that only files with a certain file extension will appear (so, for example, only image files appear). You can set the Filter property to specify a filter. For more information, see File Filtering (page 11), and for an example, see Adding a File Filter (page 18).

File Filtering

At run time when users click the **Browse** button and open the **OpenFileDialog** dialog box, they can typically select any type of file to open and no filter options will appear in the **OpenFileDialog** dialog box. You may want to limit the types of files that users can see and select in the **OpenFileDialog** dialog box, for example you might users to only be able to select image files or specific document types. You can customize the file types that users can select by setting the Filter property to a specific filter.

Basically, you can apply a filter in the **OpenFileDialog** dialog box so that only files with a certain file extension will appear. The Filter property functions similarly to the FileDialog.Filter property. You would need to set the Filter property to a filter string. For each filtering option you implement, the filter string contains a description of the filter, followed by the vertical bar (|) and the filter pattern. The strings for different filtering options are separated by the vertical bar.

The following is an example of a filter string:

- Text files (*.txt)|*.txt|All files (*.*)|*.*

You can add several filter patterns to a filter by separating the file types with semicolons, for example:

- Image Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*

You can use the FilterIndex property to set which filtering option is shown first to the user. By default the FilterIndex property is set to "1" and the first filter listed in the filter logic appears first. For example, if you include options for an image filter and for all files, as in the example above, you can show the option for all files first (and have the dialog box appear unfiltered) by setting the FilterIndex property to "2".

For an example of filtering files, see Adding a File Filter (page 18).

Working with FilePicker for WPF

ComponentOne FilePicker™ for WPF provides a simple and reliable way to select files in your WPF application. Similar to a combo box in appearance, the **C1FilePicker** control displays a file picker dialog box instead of a drop-down list-down list when the ellipses button is clicked.

Basic Properties

ComponentOne FilePicker for WPF includes several properties that allow you to set the functionality of the C1FilePicker control. Some of the more important properties are listed below.

The following properties let you customize the C1FilePicker control:

Property	Description
BrowseContent	Gets or sets the content of the **Browse** button.
DisabledCuesVisibility	Gets a value indicating whether the disabled visuals of the control are visible.
Filter	Gets or sets the filter that will be applied to the **OpenFileDialog** dialog box. See Adding a File Filter (page 18) for more information.
FilterIndex	Gets or sets the filter index that will be applied to the **OpenFileDialog** dialog box.

FocusCuesVisibility	Gets a value indicating whether the focus visuals of the control are visible.
HasSelectedFiles	True, if files were selected.
Multiselect	Gets or sets whether it's possible to select more than one file. See Selecting Multiple Files (page 17) for more information.
SelectedFile	Gets the file that the user has selected.
SelectedFiles	Gets the files that the user has selected.
SelectionBackground	Gets or sets the brush that fills the background of the selected text.
SelectionForeground	Gets or sets the brush used for the selected text in the **C1FilePicker**.
TextAlignment	Gets or sets how the text should be aligned in the **C1FilePicker**. See Changing the Text Alignment (page 18) for an example.
Watermark	Gets or sets the watermark content. See Watermark Text (page 9) for more information and Removing the Watermark (page 16) for an example.

Basic Methods

ComponentOne FilePicker for WPF includes several methods that allow you to customize the control. Some of the more important methods are listed below.

The following methods let you customize the C1FilePicker control:

Method	Description
ClearSelection	Removes the selected files. See Clearing Selected Files (page 17) for more information.
OnSelectedFilesChanged	Raises the SelectedFilesChanged event.

Basic Events

ComponentOne FilePicker for WPF includes several 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 C1FilePicker control:

Event	Description
SelectedFilesChanged	Fires when the SelectedFile property changes.

FilePicker Layout and Appearance

The following topics detail how to customize the C1FilePicker 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 WPF's XAML-based styling. You can also use templates to format and lay out the control and to customize the control's actions.

FilePicker Appearance Properties

ComponentOne FilePicker for WPF 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.

Content Properties

The following properties let you customize the appearance of content in the **C1FilePicker** control:

Property	Description
Watermark	Gets or sets the content of the watermark. See Watermark Text (page 9) for more information and Removing the Watermark (page 16) for an example.

Text Properties

The following properties let you customize the appearance of text in the **C1FilePicker** 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.
TextAlignment	Gets or sets how the text should be aligned in the **C1FilePicker** control.

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.
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 **C1FilePicker** 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.

ComponentOne ClearStyle Technology

ComponentOne ClearStyle™ technology is a new, quick and easy approach to providing Silverlight and WPF control styling. ClearStyle allows you to create a custom style for a control without having to deal with the hassle of XAML templates and style resources.

Currently, to add a theme to all standard WPF controls, you must create a style resource template. In Microsoft Visual Studio this process can be difficult; this is why Microsoft introduced Expression Blend to make the task a bit easier. Having to jump between two environments can be a bit challenging to developers who are not familiar with Blend or do not have the time to learn it. You could hire a designer, but that can complicate things when your designer and your developers are sharing XAML files.

That's where ClearStyle comes in. With ClearStyle the styling capabilities are brought to you in Visual Studio in the most intuitive manner possible. In most situations you just want to make simple styling changes to the controls in your application so this process should be simple. For example, if you just want to change the row color of your data grid this should be as simple as setting one property. You shouldn't have to create a full and complicated-looking template just to simply change a few colors.

How ClearStyle Works

Each key piece of the control's style is surfaced as a simple color property. This leads to a unique set of style properties for each control. For example, a **Gauge** has **PointerFill** and **PointerStroke** properties, whereas a **DataGrid** has **SelectedBrush** and **MouseOverBrush** for rows.

Let's say you have a control on your form that does not support ClearStyle. You can take the XAML resource created by ClearStyle and use it to help mold other controls on your form to match (such as grabbing exact colors). Or let's say you'd like to override part of a style set with ClearStyle (such as your own custom scrollbar). This is also possible because ClearStyle can be extended and you can override the style where desired.

ClearStyle is intended to be a solution to quick and easy style modification but you're still free to do it the old fashioned way with ComponentOne's controls to get the exact style needed. ClearStyle does not interfere with those less common situations where a full custom design is required.

ClearStyle Properties

The following table lists all of the ClearStyle-supported properties in the C1FilePicker control as well as a description of the property:

Property	Description
Background	Gets or sets a brush that describes the background of a control.
BorderBrush	Gets or sets a brush that describes the border background of a control.
ButtonBackground	Gets or sets the brush that will be assigned to the background of the buttons inside the control.
ButtonForeground	Gets or sets the brush that will be assigned to the foreground of the buttons inside the control.
FocusBrush	Gets or sets the brush used to highlight the focused control.
Foreground	Gets or sets a brush that describes the foreground color.
MouseOverBrush	Gets or sets the brush used to highlight the control when it has the mouse over.
PressedBrush	Gets or sets the brush used to paint a button when it is pressed.
SelectionBackground	Gets or sets the brush that fills the background of the selected text.
SelectionForeground	Gets or sets the brush used as foreground of the selected text.

FilePicker for WPF Samples

ComponentOne FilePicker for WPF includes C# samples. By default samples are installed in the **Documents** or **My Documents** folder in the **ComponentOne Samples\Studio for WPF** folder.

The following sample is included:

Sample	Description

Input Form	This sample demonstrates how to use the **C1FilePicker** component and shows how the **C1FilePicker** control can be used to create an input form. This page is part of the **C1_Demo** sample and is installed by default at **C1.WPF\C1_Demo\C1_Demo\C1Inputs\DemoInputs.xaml** in the samples directory.

FilePicker for WPF 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 C1FilePicker control in general. If you are unfamiliar with the **ComponentOne FilePicker for WPF** product, please see the FilePicker for WPF Quick Start (page 3) first.

Each topic in this section provides a solution for specific tasks using the **ComponentOne FilePicker for WPF** product. Most task-based help topics also assume that you have created a new WPF project and added the C1FilePicker control to the application.

Removing the Watermark

The watermark appears in the text area of the C1FilePicker control by default and can give directions or suggestions for users at run time. For more information, see the Watermark Text (page 9) topic. You can customize the watermark text by setting the Watermark property to the value you want to appear.

If you do not want a watermark to appear, you can set the Watermark property to a blank value, for example see the steps below.

At Design Time

To set the Watermark property at design time, complete the following steps:

1. Click the C1FilePicker control once to select it.
2. Navigate to the Properties window and locate the **Watermark** item.
3. Click in the text box next to the **Watermark** item, and delete the current text.

   This will set the Watermark property to a blank value.

In XAML

For example, to set the Watermark property add `Watermark=""` to the `<c1:C1FilePicker>` tag so that it appears similar to the following:

```
<c1:C1FilePicker HorizontalAlignment="Left" Margin="112,36,0,0"
Name="C1FilePicker1" VerticalAlignment="Top" Width="161" Watermark="" />
```

In Code

For example, to set the Watermark property, add the following code to your project:

- Visual Basic

  ```
  Me.C1FilePicker1.Watermark = ""
  ```

- C#

  ```
  this.c1FilePicker1.Watermark = "";
  ```

What You've Accomplished

You've removed the default watermark from the C1FilePicker control. Run the application, and observe that the caption bar of the C1FilePicker control will appear without the watermark:

Clearing Selected Files

When a user selects a file at run time using the C1FilePicker control, the file name appears in the text area of the control. If you want to clear all the files selected by the C1FilePicker and remove the selected file names from the control, you can use the ClearSelection method. In this example, you'll add a button to your application to clear the C1FilePicker control.

For example, add a button to your application and in the button's **Click** event handler add the following code:

- Visual Basic
```
C1FilePicker1.ClearSelection()
```
- C#
```
c1FilePicker1.ClearSelection();
```

What You've Accomplished

Run the application, and at run time choose a file in the C1FilePicker control. Click the button that you added and observe that the C1FilePicker control's file selection is cleared.

Selecting Multiple Files

FilePicker for WPF allows users to select multiple files at run time, though this option is not selected by default. At run time, users can only select one file at a time by default – you can allow users to select multiple files by setting the Multiselect property to **True**, for example see the steps below.

At Design Time

To set the Multiselect property at design time, complete the following steps:

1. Click the C1FilePicker control once to select it.
2. Navigate to the Properties window and locate the **Multiselect** item.
3. Check the check box next to the **Multiselect** item.

   This will set the Multiselect property to allow users to select multiple files at run time in the **OpenFileDialog** dialog box.

In XAML

For example, to set the Multiselect property add `Multiselect="True"` to the `<c1:C1FilePicker>` tag so that it appears similar to the following:

```
<c1:C1FilePicker HorizontalAlignment="Left" Margin="112,36,0,0"
Name="C1FilePicker1" VerticalAlignment="Top" Width="161" Multiselect="True"
/>
```

In Code

For example, to set the Multiselect property, add the following code to your project:

- Visual Basic
```
Me.C1FilePicker1.Multiselect = True
```
- C#
```
this.c1FilePicker1.Multiselect = true;
```

What You've Accomplished

You can now select multiple files using the C1FilePicker control. Run the application, select the **Browse** button in the C1FilePicker control, and observe that you can select multiple files in the dialog box that appears by clicking the CTRL key while choosing files.

Changing the Text Alignment

The C1FilePicker control typically displays the current file selected in the text area of the control. This text is aligned to the left by default, but you can change the text alignment if you choose. Text alignment can be set using the TextAlignment property, and options include **Left** (default), **Center**, or **Right** text alignment. For more information, see the steps below.

At Design Time

To set the TextAlignment property to **Right** at design time, complete the following steps:

1. Click the C1FilePicker control once to select it.
2. Navigate to the Properties window and locate the **TextAlignment** item.
3. Click the drop-down arrow next to the **TextAlignment** item and select **Right**.

In XAML

For example, to set the TextAlignment property to **Right** add `TextAlignment="Right"` to the `<c1:C1FilePicker>` tag so that it appears similar to the following:

```
<c1:C1FilePicker HorizontalAlignment="Left" Margin="112,36,0,0"
Name="C1FilePicker1" VerticalAlignment="Top" Width="161"
TextAlignment="Right" />
```

In Code

For example, to set the TextAlignment property to **Right**, add the following code to your project:

- Visual Basic

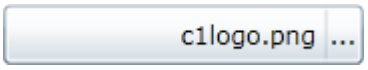
  ```
  Me.C1FilePicker1.TextAlignment = TextAlignment.Right
  ```

- C#

  ```
  this.c1FilePicker1.TextAlignment = TextAlignment.Right;
  ```

What You've Accomplished

Text is now aligned to the right. Run the application and select a file. Notice that the name of the file is now aligned to the right in the C1FilePicker control:



Adding a File Filter

At run time when users click the **Browse** button and open the **OpenFileDialog** dialog box, they can typically select any type of file to open. You can apply a filter in the **OpenFileDialog** dialog box so that only files with a certain file extension will appear. So, for example, you can specify a filter so that only image files appear or only text files appear. You can set the Filter property to specify a filter as in the following steps.

At Design Time

To set the Filter property to filter image files at design time, complete the following steps:

1. Click the C1FilePicker control once to select it.
2. Navigate to the Properties window and locate the **Filter** item.

3. In the text box next to the **Filter** item and enter "Image Files(*.PNG;*.JPG;*.GIF)|*.PNG;*.JPG;*.GIF|All files (*.*)|*.*".

In XAML

For example, to set the Filter property to filter image files add `Filter="Image Files(*.PNG;*.JPG;*.GIF)|*.PNG;*.JPG;*.GIF|All files (*.*)|*.*"` to the `<c1:C1FilePicker>` tag so that it appears similar to the following:

```
<c1:C1FilePicker HorizontalAlignment="Left" Margin="112,36,0,0"
Name="C1FilePicker1" VerticalAlignment="Top" Width="161" Filter="Image
Files(*.PNG;*.JPG;*.GIF)|*.PNG;*.JPG;*.GIF|All files (*.*)|*.*" />
```

In Code

For example, to set the Filter property to filter image files, add the following code to your project:

- Visual Basic

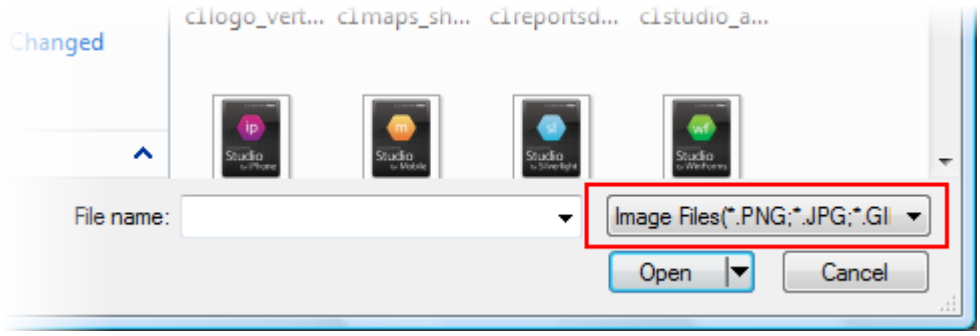
```
Me.C1FilePicker1.Filter = "Image
Files(*.BMP;*.JPG;*.GIF)|*.PNG;*.JPG;*.GIF|All files (*.*)|*.*"
```

- C#

```
this.c1FilePicker1.Filter = "Image
Files(*.BMP;*.JPG;*.GIF)|*.PNG;*.JPG;*.GIF|All files (*.*)|*.*";
```

What You've Accomplished

A filter will be applied to the **OpenFileDialog** dialog box. Run the application and click the **Browse** button in the C1FilePicker control. Notice that a filter has been applied so that only files with the PNG, JPG, and GIF extensions appear in the dialog box:



If you click the filter box's drop-down arrow you can choose to display all files. That is because the "All files" option was included in the filter logic applied in the steps above.