

---

ComponentOne

# MediaPlayer for WPF

Copyright © 2012 ComponentOne LLC. All rights reserved.

*Corporate Headquarters*  
**ComponentOne LLC**  
201 South Highland Avenue  
3<sup>rd</sup> Floor  
Pittsburgh, PA 15206 • USA

**Internet:** [info@ComponentOne.com](mailto:info@ComponentOne.com)

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

**Sales**

E-mail: [sales@componentone.com](mailto:sales@componentone.com)

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

**Trademarks**

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

**Warranty**

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

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

**Copying and Distribution**

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

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

# Table of Contents

ComponentOne MediaPlayer for WPF .....	1
Installing MediaPlayer for WPF .....	1
Studio for WPF Setup Files.....	1
Using Maps Powered by Esri .....	2
System Requirements .....	3
Installing Demonstration Versions .....	4
Uninstalling MediaPlayer for WPF.....	4
End-User License Agreement .....	4
Licensing FAQs .....	4
What is Licensing?.....	4
How does Licensing Work?.....	4
Common Scenarios .....	5
Troubleshooting.....	7
Technical Support .....	9
Redistributable Files.....	9
About this Documentation.....	10
XAML and XAML Namespaces.....	10
Creating a Microsoft Blend Project.....	11
Creating a .NET Project in Visual Studio .....	12
Creating an XAML Browser Application (XBAP) in Visual Studio .....	13
Adding the MediaPlayer for WPF Components to a Blend Project .....	14
Adding the MediaPlayer for WPF Components to a Visual Studio Project.....	14
Key Features .....	15
MediaPlayer for WPF Quick Start .....	16
Step 1 of 3: Creating an Application with a C1MediaPlayer Control.....	16
Step 2 of 3: Adding Content to the C1MediaPlayer Control .....	16
Step 3 of 3: Running the Project .....	17
Using C1MediaPlayer.....	19
C1MediaPlayer Elements.....	20
Screen .....	21

Item List .....	22
Chapter List .....	23
Adjust Volume.....	24
Time Presenter.....	24
Supported File Types .....	24
MediaPlayer for WPF Layout and Appearance .....	25
ComponentOne ClearStyle Technology .....	25
How ClearStyle Works.....	26
C1MediaPlayer ClearStyle Properties.....	26
MediaPlayer for WPF Appearance Properties.....	28
Text Properties .....	28
Color Properties.....	28
Border Properties.....	28
Size Properties.....	28
Templates.....	29
MediaPlayer for WPF Samples.....	30
MediaPlayer for WPF Task-Based Help.....	30
Adding Media Content .....	30
Creating Chapters.....	32
Turning Off Autoplay .....	33
Looping Media Files.....	33
Setting the Initial Volume .....	34
Showing the Chapter List on Page Load .....	35
Showing the Item List on Page Load.....	35

# ComponentOne MediaPlayer for WPF

**ComponentOne MediaPlayer™ for WPF** provides a player that runs all video and audio media formats supported by WPF. Implement advanced capabilities with ease such as playlists, full-screen mode, overlay support, and more.

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



## Getting Started

- [Using C1MediaPlayer](#) (page 19)
- [Quick Start](#) (page 16)
- [Task-Based Help](#) (page 30)

## Installing MediaPlayer for WPF

The following sections provide helpful information on installing **ComponentOne MediaPlayer for WPF**.

### Studio for WPF Setup Files

The installation program will create the directory **C:\Program Files\ComponentOne\Studio for WPF**, which contains the following subdirectories:

#### Bin

Contains copies of all ComponentOne binaries (DLLs, EXEs). For **Component MediaPlayer for WPF**, the following DLLs are installed:

- C1.WPF.dll
- C1.WPF.Expression.Design.dll
- C1.WPF.Expression.Design.4.0.dll
- C1.WPF.VisualStudio.Design.dll
- C1.WPF.VisualStudio.Design.4.0.dll
- C1.WPF.MediaPlayer.dll
- C1.WPF.MediaPlayer.Expression.Design.dll
- C1.WPF.MediaPlayer.Expression.Design.4.0.dll
- C1.WPF.MediaPlayer.VisualStudio.Design.dll
- C1.WPF.MediaPlayer.VisualStudio.Design.4.0.dll

In addition, the following files from the Microsoft WPF Toolkit are also installed:

- WPFToolkit.dll
- WPFToolkit.Design.dll
- WPFToolkit.VisualStudio.Design.dll

For more information about the Microsoft WPF Toolkit, see [CodePlex](#). The C1.WPF.dll and WPFToolkit.dll assemblies are required for deployment.

<b>HelpViewer</b>	Contains online documentation for all Studio components.
<b>C1WPF\XAML</b>	Contains the full XAML definitions of C1MediaPlayer styles and templates which can be used for creating your own custom styles and templates.

## Samples

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

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

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

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

<b>Common</b>	Contains support and data files that are used by many of the demo programs.
<b>Studio for WPF</b>	Contains samples for <b>Accordion for WPF</b> .

Samples can be accessed from the **ComponentOne Control Explorer**. To view samples, on your desktop, click the **Start** button and then click **ComponentOne | Studio for WPF | Samples | WPF ControlExplorer**.

## Esri Maps

Esri® files are installed with **ComponentOne Studio for Silverlight**, **ComponentOne Studio for WPF**, and **ComponentOne Studio for Windows Phone** by default to the following folders:

32-bit machine : C:\Program Files\ESRI SDKs\\<version number>

64-bit machine: C:\Program Files (x86)\ESRI SDKs\\<version number>

Files are provided for multiple languages, including: English, German (de), Spanish (es), French (fr), Italian (it), Japanese (ja), Portuguese (pt-BR), Russian (ru) and Chinese (zh-CN).

See [Using Maps Powered by Esri](#) (page 2) or visit the Esri website at <http://www.esri.com> for additional information.

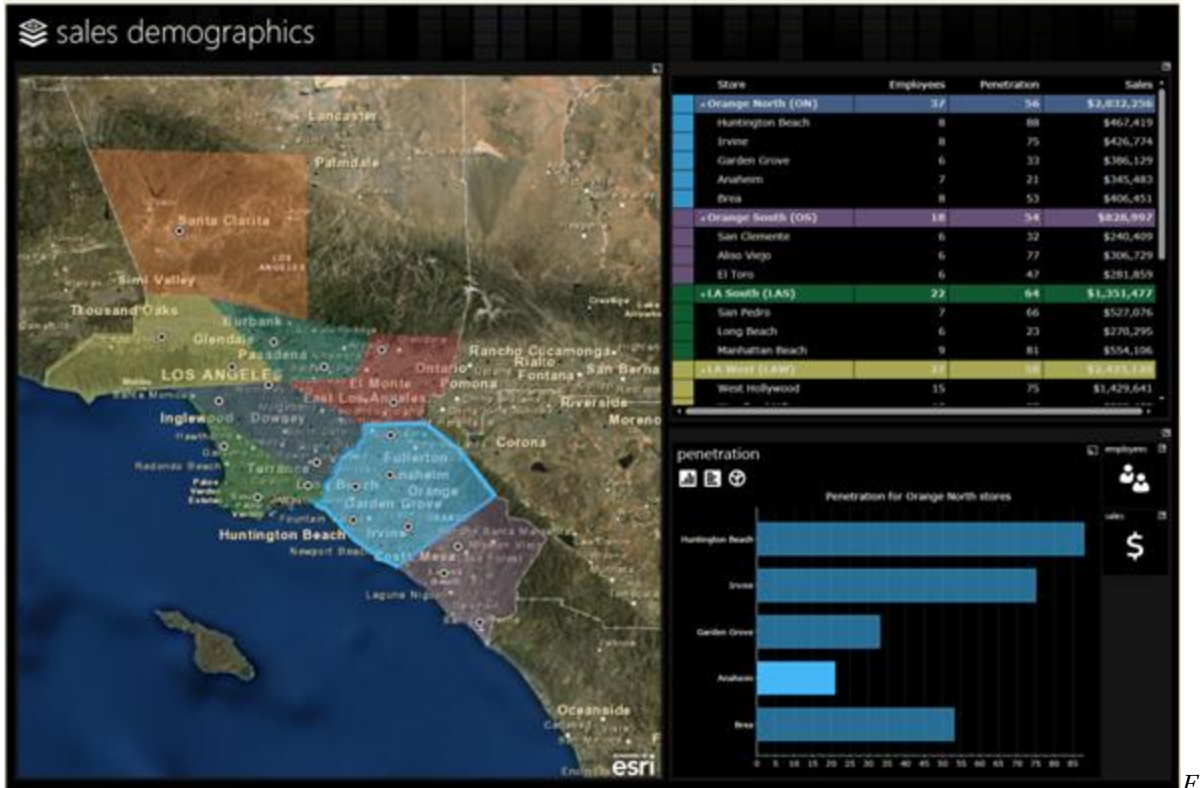
## Using Maps Powered by Esri

Easily transform GIS data into business intelligence with controls for Silverlight, WPF, and Windows Phone powered by Esri® software.

By using the ComponentOne award-winning UI controls, you'll have the tools you need to seamlessly create rich, map-enabled user interfaces.

Benefits of Maps powered by Esri:

- Esri knows maps: Esri is the leading online map and GIS provider.
- Maps are technical: Using maps within your application is a very technical thing, so you don't want to take your chance using anyone but the best.
- Company of choice: Esri is the company of choice of many top companies and government agencies.
- Fulfill any developers' mapping needs: Esri mapping tools are flexible and will fill the needs of any mapping solution.



sri Map Example

There are no additional charges for using the Esri maps included with ComponentOne products. Simply create a free online account at <http://www.arcgisonline.com> to start taking advantage of the Esri map controls. Esri licensing terms can be found in our Licensing Information and End User Licensing Agreement at <http://www.componentone.com/SuperPages/Licensing/>.

To learn more about Esri and Esri maps, please visit Esri at <http://www.esri.com>. There you will find detailed support, including [documentation](#), [forums](#), [samples](#), and much more.

See the [Studio for WPF Setup Files](#) (page 1) topic for more information on the Esri files installed with this product.

## System Requirements

System requirements include the following:

**Operating Systems:** Microsoft Windows® XP with Service Pack 2 (SP2)  
 Windows Vista™  
 Windows 7  
 Windows 2008 Server

**Environments:** .NET Framework 3.5 or later  
 Visual Studio® 2005 extensions for .NET Framework 2.0  
 November 2006 CTP  
 Visual Studio® 2008 or later

**Microsoft® Expression®  
Blend Compatibility:**

**MediaPlayer for WPF** includes design-time support for Expression Blend.

**Note:** The **C1.WPF.VisualStudio.Design.dll** assembly is required by Visual Studio 2008 and the **C1.WPF.Expression.Design.dll** assembly is required by Expression Blend. The **C1.WPF.Expression.Design.dll** and **C1.WPF.VisualStudio.Design.dll** assemblies installed with **MediaPlayer for WPF** should always be placed in the same folder as **C1.WPF.dll**; the DLLs should NOT be placed in the Global Assembly Cache (GAC).

## Installing Demonstration Versions

If you wish to try **ComponentOne MediaPlayer for WPF** 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 registered versions will stamp every application you compile so that a ComponentOne banner will not appear when your users run the applications.

## Uninstalling MediaPlayer for WPF

To uninstall **ComponentOne MediaPlayer for WPF**:

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

## End-User License Agreement

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

## Licensing FAQs

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

### What is Licensing?

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

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

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

### How does Licensing Work?

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

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

When a user decides to purchase a product, he receives an installation program and a Serial Number. During the installation process, the user is prompted for the serial number that is saved on the system. (Users can also enter the serial number by clicking the **License** button on the **About Box** of any ComponentOne product, if available, or by rerunning the installation and entering the serial number in the licensing dialog box.)

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

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

These files are automatically added to the project.

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

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

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

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

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

## Common Scenarios

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

### *Creating components at design time*

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

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

### *Creating components at run time*

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

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

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

Adding an instance of the component to a form, then removing that component, is just a simple way of adding a line with the component strong name to the **licenses.licx** file. If desired, you can do this manually using notepad or Visual Studio itself by opening the file and adding the text. When Visual Studio recreates the application resources, the component will be queried and its run-time license added to the appropriate assembly resource.

### ***Inheriting from licensed components***

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

- Add a **LicenseProvider** attribute to the component.

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

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

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

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

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

### ***Using licensed components in console applications***

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

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

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

### ***Using licensed components in Visual C++ applications***

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

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

1. Build the C++ project as usual. This should create an EXE file and also a licenses.licx file with licensing information in it.
2. Copy the **licenses.licx** file from the application directory to the target folder (**Debug** or **Release**).

3. Copy the **C1Lc.exe** utility and the licensed DLLs to the target folder. (Don't use the standard `lc.exe`, it has bugs.)
4. Use **C1Lc.exe** to compile the **licenses.licx** file. The command line should look like this:  
`c1lc /target:MyApp.exe /complist:licenses.licx /i:C1.Win.C1FlexGrid.dll`
5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select **Properties**, and go to the **Linker/Command Line** option. Enter the following:  
`/ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses`
6. Rebuild the executable to include the licensing information in the application.

### **Using licensed components with automated testing products**

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

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

For example:

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

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

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

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

### **Troubleshooting**

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

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

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

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

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

#### **If that fails follow these steps:**

1. Open the affected project.
2. Select an instance of the updated component.
3. In the Visual Studio Properties window, change any property. It does not matter which property you change; you can change it back to the previous value.

4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

**Alternative 1: Follow these steps:**

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

**Alternative 2: Follow these steps:**

1. Open the affected project.
2. Delete the licenses.licx file from the project.
3. Add a new instance of the updated component to the form.
4. Rebuild and run the solution. The nag screen should not appear.
5. Remove the newly added component from the form.
1. Try each of these options multiple times, if necessary. If that still does not help, are you creating any of the controls in code rather than design-time? If so, you must add an entry for the control in the licenses.licx file (see <http://helpcentral.componentone.com/PrintableView.aspx?ID=1886> for more information). Also if this is a Web site, as opposed to an ASP.NET Web application, please try right-clicking the licenses.licx file and selecting "Build Runtime Licenses" from the context menu.

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

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

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

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

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

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

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

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

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

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

## Option 2 – Continue to use the components you have.

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

## Technical Support

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

Some methods for obtaining technical support include:

- **[Online Resources](#)**  
ComponentOne provides customers with a comprehensive set of technical resources in the form of FAQs, [samples and videos](#), Version Release History, searchable Knowledge base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**  
This online support service provides you with direct access to our Technical Support staff via an [online incident submission form](#). When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.
- **Peer-to-Peer Product Forums**  
ComponentOne peer-to-peer product [forums](#) 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**  
Microsoft integrated ComponentOne documentation can be installed with each of our products, and documentation is also available online. If you have suggestions on how we can improve our documentation, please email the [Documentation team](#). Please note that e-mail sent to the [Documentation team](#) is for documentation feedback only. [Technical Support](#) and [Sales](#) issues should be sent directly to their respective departments

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

## Redistributable Files

**ComponentOne MediaPlayer for WPF** 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.WPF.dll
- C1.WPF.MediaPlayer.dll

In addition, the following file from the Microsoft WPF Toolkit is also installed and is redistributable:

- WPFToolkit.dll

Site licenses are available for groups of multiple developers. Please contact [Sales@ComponentOne.com](mailto:Sales@ComponentOne.com) for details.

## About this Documentation

You can create your applications using Microsoft Expression Blend or Visual Studio, but Blend is currently the only design-time environment that allows users to design XAML documents visually. In this documentation, we will use the **Design** workspace of Blend for most examples.

### Acknowledgements

*Microsoft, Windows, Windows Vista, Visual Studio, and Microsoft Expression are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Esri is a registered trademark of Environmental Systems Research Institute, Inc. (Esri) in the United States, the European Community, or certain other jurisdictions.*

### 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](#).

## XAML and XAML Namespaces

XAML is a declarative XML-based language that is used as a user interface markup language in Windows Presentation Foundation (WPF) and the .NET Framework 3.0. With XAML you can create a graphically rich customized user interface, perform data binding, and much more. For more information on XAML and the .NET Framework 3.0, please see <http://www.microsoft.com>.

### XAML Namespaces

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

When you create a Microsoft Expression Blend project, a XAML file is created for you and some initial namespaces are specified:

Namespace	Description
xmlns="http://schemas.microsoft.com/win	This is the default Windows Presentation Foundation

fx/2006/xaml/presentation"	namespace.
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"	This is a XAML namespace that is mapped to the <b>x:</b> prefix. The <b>x:</b> prefix provides a quick, easy way to reference the namespace, which defines many commonly-used features necessary for WPF applications.

When you add a `ClMediaPlayer` control to the window in Microsoft Expression Blend or Visual Studio, **Blend** or **Visual Studio** automatically creates an XML namespace for the control. The namespace looks like the following:

```
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml">
```

The namespace value is `c1`. This is a unified namespace; once this is in the project, all ComponentOne WPF controls found in your references will be accessible through XAML (and Intellisense). Note that you still need to add references to the assemblies for each control you need to use.

You can also choose to create your own custom name for the namespace. For example:

```
xmlns:MyMTB=http://schemas.componentone.com/wpf/ClMediaPlayer
```

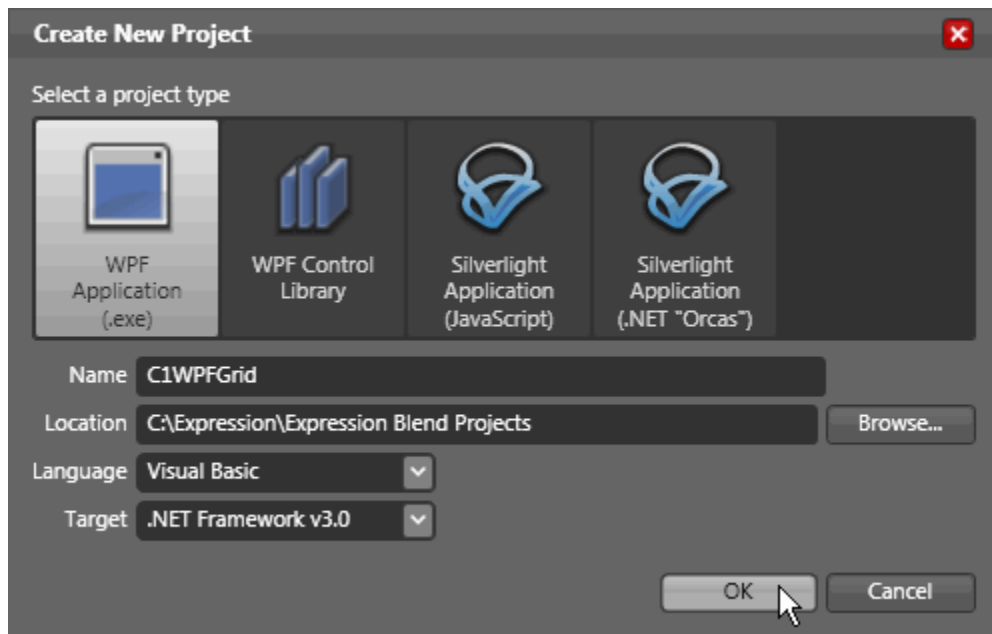
You can now use your custom namespace when assigning properties, methods, and events. For example, use the following XAML to add a border around the panel:

```
<MyMTB:ClMediaPlayer Name="c1MediaPlayer1" BorderThickness="10,10,10,10">
```

## Creating a Microsoft Blend Project

To create a new Blend project, complete the following steps:

1. From the **File** menu, select **New Project** or click **New Project** in the Blend startup window.  
The **Create New Project** dialog box opens.
2. Make sure **WPF Application (.exe)** is selected and enter a name for the project in the Name text box. The **WPF Application (.exe)** creates a project for a Windows-based application that can be built and run while being designed.
3. Select the **Browse** button to specify a location for the project.
4. Select a language from the **Language** drop-down box and click **OK**.

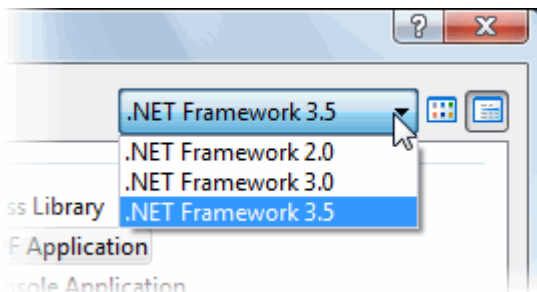


A new Blend project with a XAML window is created.

## Creating a .NET Project in Visual Studio

To create a new .NET project in Visual Studio 2008, complete the following steps:

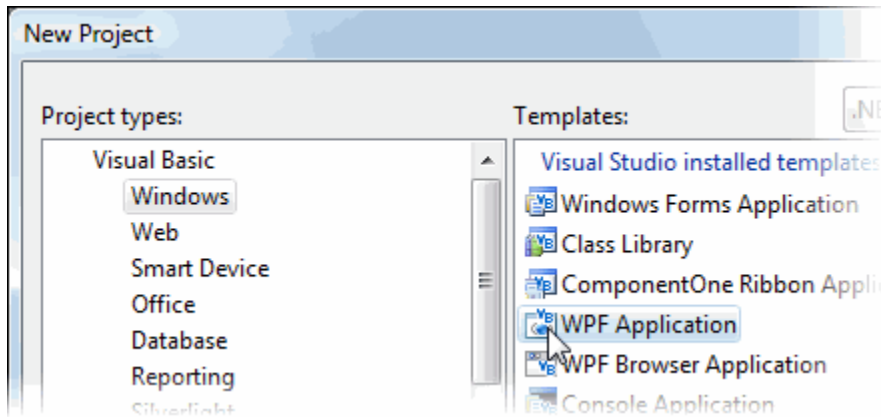
1. From the **File** menu in Microsoft Visual Studio 2008, select **New Project**.  
The **New Project** dialog box opens.
2. Choose the appropriate .NET Framework from the Framework drop-down box in the top-right of the dialog box.



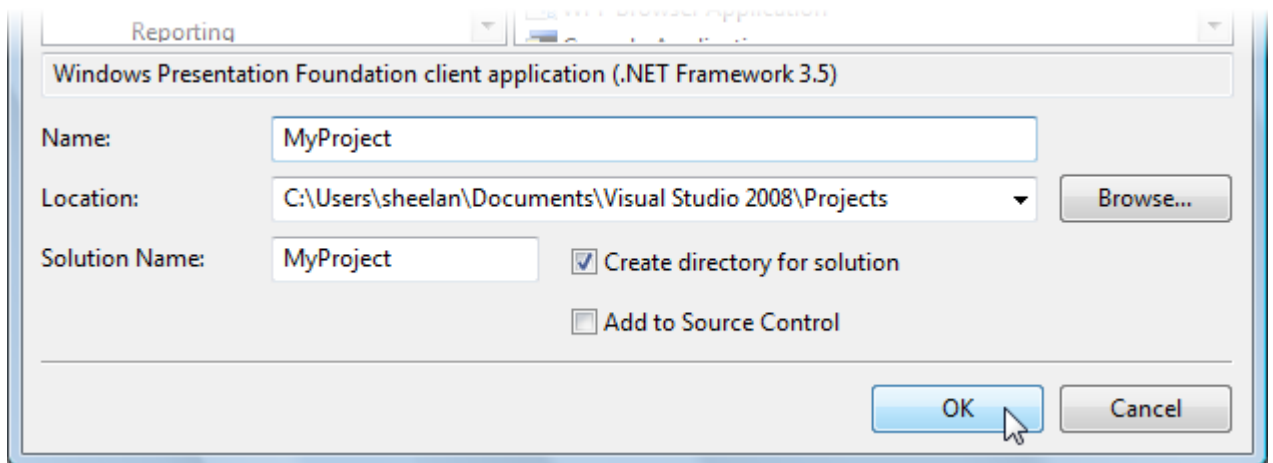
3. Under **Project types**, select either **Visual Basic** or **Visual C#**.

**Note:** In Visual Studio 2005 select **NET Framework 3.0** under **Visual Basic** or **Visual C#** in the Project types menu.

4. Choose **WPF Application** from the list of **Templates** in the right pane.



5. Enter a name for your application in the **Name** field and click **OK**.



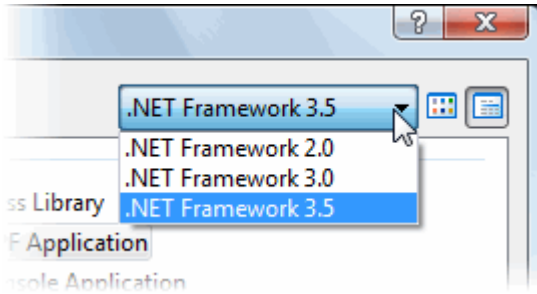
A new Microsoft Visual Studio .NET WPF project is created with a XAML file that will be used to define your user interface and commands in the application.

**Note:** You can create your WPF applications using Microsoft Expression Blend or Visual Studio, but Blend is currently the only design-time environment that allows users to design XAML documents visually. In this documentation, Blend will be used for most examples.

## Creating an XAML Browser Application (XBAP) in Visual Studio

To create a new XAML Browser Application (XBAP) in Visual Studio 2008, complete the following steps:

1. From the **File** menu in Microsoft Visual Studio 2008, select **New Project**. The **New Project** dialog box opens.
2. Choose the appropriate .NET Framework from the Framework drop-down box in the top-right of the dialog box.



3. Under Project types, select either **Visual Basic** or **Visual C#**.
4. Choose **WPF Browser Application** from the list of **Templates** in the right pane.

**Note:** If using Visual Studio 2005, you may need to select **XAML Browser Application (WPF)** after selecting **NET Framework 3.0** under **Visual Basic** or **Visual C#** in the left-side menu.

5. Enter a name for your application in the **Name** field and click **OK**.

A new Microsoft Visual Studio .NET WPF Browser Application project is created with a XAML file that will be used to define your user interface and commands in the application.

## Adding the MediaPlayer for WPF Components to a Blend Project

In order to use **C1MediaPlayer** or another **ComponentOne MediaPlayer for WPF** component in the Design workspace of Blend, you must first add a reference to the **C1.WPF.Extended** assembly and then add the component from Blend's **Asset Library**.

### To add a reference to the assembly:

1. Select Project | Add Reference.
2. Browse to find the **C1.WPF.MediaPlayer.dll** assembly installed with **MediaPlayer for WPF**.

**Note:** The **C1.WPF.MediaPlayer.dll** file is installed to *C:\Program Files\ComponentOne\Studio for WPF\bin* by default.

3. Select **C1.WPF.MediaPlayer.dll** and click **Open**. A reference is added to your project.

### To add a component from the Asset Library:

1. Once you have added a reference to the **C1.WPF.MediaPlayer** assembly, click the **Asset Library** button



in the Blend Toolbox. The **Asset Library** appears:

2. Expand the **Controls** node and select **All**.
3. Select **C1MediaPlayer**. The component will appear in the Toolbox above the **Asset Library** button.
4. Double-click the **C1MediaPlayer** component in the Toolbox to add it to **Window1.xaml**.

## Adding the MediaPlayer for WPF Components to a Visual Studio Project

When you install **ComponentOne MediaPlayer for WPF** the **C1MediaPlayer** control should be added to your Visual Studio Toolbox. You can also manually add ComponentOne controls to the Toolbox.

**ComponentOne MediaPlayer for WPF** provides the following control:

- C1MediaPlayer

To use a **MediaPlayer for WPF** panel or control, add it to the window or add a reference to the **C1.WPF** assembly to your project.

### Manually Adding MediaPlayer for WPF to the Toolbox

When you install **MediaPlayer for WPF**, the following **MediaPlayer for WPF** control and panel will appear in the Visual Studio Toolbox customization dialog box:

- C1MediaPlayer

To manually add the C1MediaPlayer control to the Visual Studio 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 **MediaPlayer for WPF** components appear on its own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **C1WPFMediaPlayer**, for example.
3. Right-click the tab where the component is to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
4. In the dialog box, select the **WPF Components** tab.
5. Sort the list by Namespace (click the *Namespace* column header) and select the check boxes for components belonging to the **C1.WPF.MediaPlayer** namespace. Note that there may be more than one component for each namespace.

### Adding MediaPlayer for WPF to the Window

To add **ComponentOne MediaPlayer for WPF** to a window or page, complete the following steps:

1. Add the C1MediaPlayer control to the Visual Studio Toolbox.
2. Double-click C1MediaPlayer or drag the control onto the window.

### Adding a Reference to the Assembly

To add a reference to the **MediaPlayer for WPF** assembly, complete the following steps:

1. Select the **Add Reference** option from the **Project** menu of your project.
2. Select the **ComponentOne MediaPlayer for WPF** assembly from the list on the **.NET** tab or on the **Browse** tab, browse to find the **C1.WPF.dll** assembly and click **OK**.
3. Double-click the window caption area to open the code window. At the top of the file, add the following **Imports** statements (**using** in C#):

```
Imports C1.WPF.MediaPlayer
```

This makes the objects defined in the **MediaPlayer for WPF** assembly visible to the project.

# Key Features

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

- **Create Playlists**  
Add your own playlist in line or in the code behind.
- **Full Screen Mode**

Easily set the player to display in full screen mode.

- **Overlay Support**

Add an overlay to your WPF media player; for example, you can overlay logos, banners, and advertisements.

# MediaPlayer for WPF Quick Start

The following quick start guide is intended to get you up and running with **MediaPlayer for WPF**. In this quick start, you'll start in Visual Studio to create a new project, add a C1MediaPlayer control to your application, and add chaptered content to the C1MediaPlayer control.

## Step 1 of 3: Creating an Application with a C1MediaPlayer Control

In this step, you'll begin in Visual Studio to create a WPF application using **MediaPlayer for WPF**.

Complete the following steps:



1. In Visual Studio 2008, 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 and click **OK**.
3. Click the **Split** button to enter Split view.
4. Navigate to the Toolbox and double-click the C1MediaPlayer icon to add the C1MediaPlayer control to the project.
5. Click the C1MediaPlayer control to select it.
6. In the **Properties** window, set the following properties:
  - Set the **Height** property to "300".
  - Set the **Width** property to "350".

You've successfully created a WPF application containing a C1MediaPlayer control. In the next step, you will add video content to the C1MediaPlayer control.

## Step 2 of 3: Adding Content to the C1MediaPlayer Control

In this section of the quick start tutorial, you will add a video with one chapter to the C1MediaPlayer control.

Complete the following steps:

1. In the **Properties** window, click the **Items** ellipsis button .
2. The **Collection Editor: Items** dialog box appears.
3. Click **Add** to add a C1MediaItem item to the C1MediaPlayer control.
4. In the **Properties** grid, set the following properties:
5. Set the **MediaSource** property to "<http://download.componentone.com/pub/Videos/Trevor%20Does%20Silverlight.wmv>". This sets the URL of the media source.
6. Set the **Title** property to "Trevor Does Silverlight". This specifies the title of the media.
7. Set the **NaturalDuration** property to "00:28:15". This sets the duration stamp to 28 minutes and 15 seconds.
8. Click the **Chapters** ellipsis button  to open the **CollectionEditors:Chapters** dialog box.

9. Click **Add** to add a C1MediaChapter item to the C1MediaItem item.
10. In the **Properties** grid, specify the position that the chapter begins at by setting the Position property to "00:02:48".
11. In the **Properties** grid, create a title for your chapter setting the Title property to "Setting Up a Development System".

You have successfully added a video with one chapter to the C1MediaPlayer control. In the next step, you will run the project and observe several functions of the C1MediaPlayer control.


## Step 3 of 3: Running the Project

In the last step, you added a video with one chapter to the C1MediaPlayer control. In this step, you will run the project and observe some of the run-time features of the C1MediaPlayer control.

Complete the following steps:


1. From the **Debug** menu, select **Start Debugging** to view how your application will appear at run time. Observe that the video content plays automatically and that the application resembles the following:

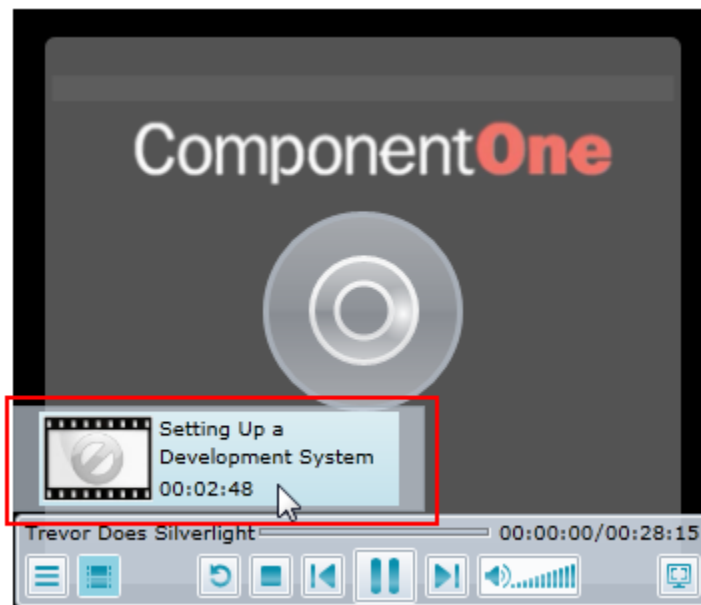


2. Click the **Item List** button  to open the item list and observe that it resembles the following:



You added the title and the duration time of the video in steps 2 and 3 of [Step 2 of 3: Adding Content to the C1MediaPlayer Control](#) (page 16).

3. Click the **Chapter List** button  to open the chapter list and observe that it resembles the following:



4. Click **Setting up a Development System** and observe that the video fast-forwards to that chapter. The media player content will appear follows:



Congratulations! You have successfully completed the **MediaPlayer for WPF** quick start. In this quick start, you've created a **MediaPlayer for WPF** application, added chaptered video content to the C1MediaPlayer control, and viewed some of the run-time capabilities of the control.

## Using C1MediaPlayer

The C1MediaPlayer control is a media device used to deliver video and audio to your users. With the C1MediaPlayer control, you can create content playlists and even separate large videos into manageable chapters. The control is able to hold any media format supported by WPF.



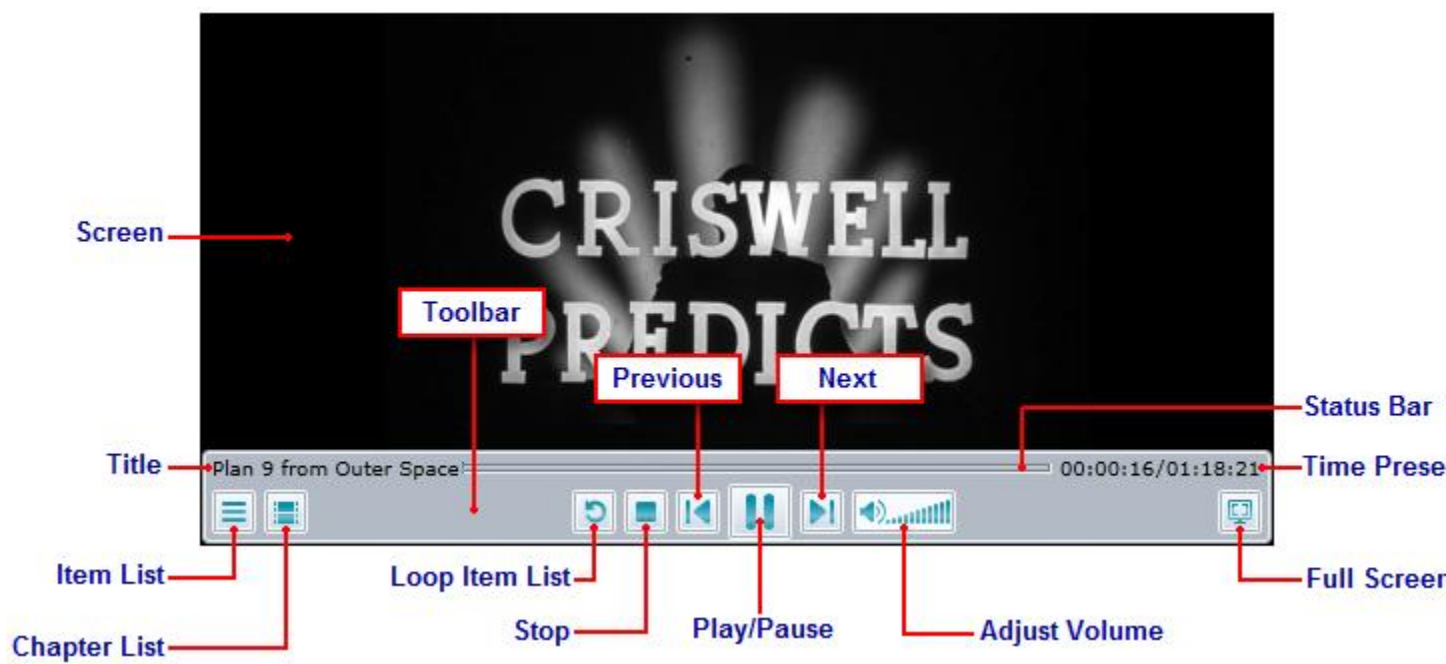
After the C1MediaPlayer control is added to your project, you can begin customizing its appearance and behaviors. You can apply themes and gradients, create video overlays, choose full screen view, add and remove buttons, and much more.

You can add content to the C1MediaPlayer control by creating C1MediaItem item and then setting its MediaSource property to the URL location of an image, video, or audio file.

To help get you started, the following topics will present you with an overview of some of the common elements and features of the C1MediaPlayer control.


## C1MediaPlayer Elements

This section provides a visual and descriptive overview of the elements that comprise the C1MediaPlayer control. The following image lists the default elements of the C1MediaPlayer control.



For a description of each C1MediaPlayer element, refer the table below.

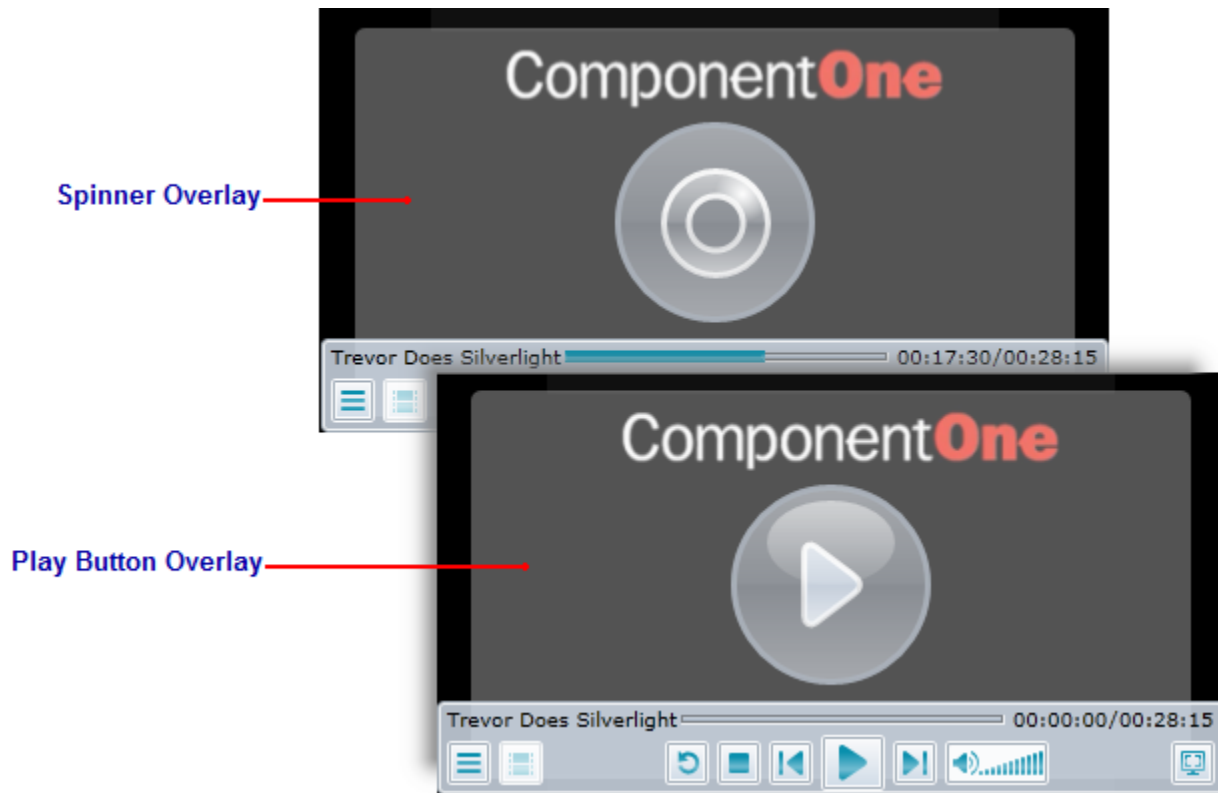
Element	Description
Screen	The C1MediaPlayer content area displays videos. For more information on the content area, see <a href="#">Screen</a> (page 21).
Toolbar	The toolbar displays all of the media player controls. The toolbar only appears at startup and when it is hovered over by a cursor.
Title	The title, which is set using the Title property, is used to display the title of a picture, video, or music file. The title is displayed by default, but you can remove it by setting the IsTitleVisible property to <b>False</b> .
Item List	The <b>Item List</b> button opens or closes the item list, which is essentially a play list that contains all of the media items added to the control. You can remove the <b>Item List</b> button by setting the IsItemListButtonVisible property to <b>False</b> .
Chapter List	The <b>Chapter List</b> button opens or closes the chapter list, which displays all of the chapters added to the current media item. You can remove the <b>Chapter List</b> button by setting the

	IsChapterListButtonVisible property to <b>False</b> .
Loop Item List	The <b>Loop Item List</b> button turns looping on or off. Looping is off by default, but you can turn on looping by setting the IsLooping property to <b>True</b> . You can remove the <b>Loop Item List</b> button by setting the IsLoopButtonVisible property to <b>False</b> .
Stop	The <b>Stop</b> button will stop the C1MediaPlayer control from playing its contents. To remove the <b>Stop</b> button, set the IsStopButtonVisible property to <b>False</b> .
Previous	The <b>Previous</b> button allows you to rewind to the preceding media file. You can remove the <b>Previous</b> button by setting the IsPreviousButtonVisible property to <b>False</b> .
Play/Pause	The <b>Play/Pause</b> button plays or pauses the open media file. The button changes depending on whether the media file is currently stopped, playing, or paused. If the media player is stopped or paused, the <b>Play</b> button will appear; if the media player is playing, the <b>Pause</b> button will appear.
Next	The <b>Next</b> button allows you to fast-forward to the subsequent media file. You can remove the <b>Next</b> button by setting the IsNextButtonVisible property to <b>False</b> .
Adjust Volume	The adjust volume slider controls the content of the media file you are currently viewing. You can click the mute button  to mute sound.
Full Screen	The <b>Full Screen</b> button opens and closes the full screen view of the C1MediaPlayer control. When the control is in full screen mode, you can return to its regular size by either clicking the <b>Full Screen</b> button again or by pressing ESC on your keyboard. To remove the <b>Full Screen</b> button, set the IsFullScreenButtonVisible property to <b>False</b> .
Time Presenter	The time presenter displays both the time elapsed and the time remaining in the current media file. You can remove the time presenter by setting the IsTimePresenterVisible property to <b>False</b> .
Status Bar	The status bar indicates both the time elapsed and the time remaining in the currently media file. It's also a slider bar that allows you to change your location in the current media file. To remove the status bar, set the IsPositionSliderVisible property to <b>False</b> .

## Screen

The screen of the C1MediaPlayer control displays videos. When the C1MediaPlayer control is playing a media file that doesn't have a visual track, such as an MP3 or a WMA file, the content area will appear with a black background.

The screen can also be covered – or at least partially covered – by several overlays. When a file is buffering, an overlay with an animated spinner will cover the content area. When a file is paused or stopped, an overlay with a play button over the content area; users can click on this overlay to begin playing the file.



The screen can also display the item list and the chapter list. The item list, which displays a list of available media files, will occupy the entire screen. The chapter list, which displays chapters within the current media file, will be contained within a rectangular overlay.

You can create a custom screen overlay by adding and then customizing the ScreenExtension template. This type of overlay can be used to display copyright information, video descriptions, or advertisements.

## Item List

The item list is a directory of the content that has been added to the media player. You can use the up or down buttons to scroll through the list and then click on one of the media files to select it. The item list looks as follows:



A few properties will have to be set to achieve the look and behavior of an item list such as the one above. To title a media item, you have to set the Title property of a C1MediaItem to a string. To add a time stamp to the video, set the NaturalDuration property. You can add the thumbnail by setting the ThumbnailSource property to the location of an image file

The following XAML was used to create the first item in this list.

```
<clmediaplayer:C1MediaItem
  MediaSource="http://www.acme.org/download/public_domain_cartoon.mp4"
  ThumbnailSource="FelixMiddle.png" NaturalDuration="00:07:58"
  Title="Felix the Cat" >
```

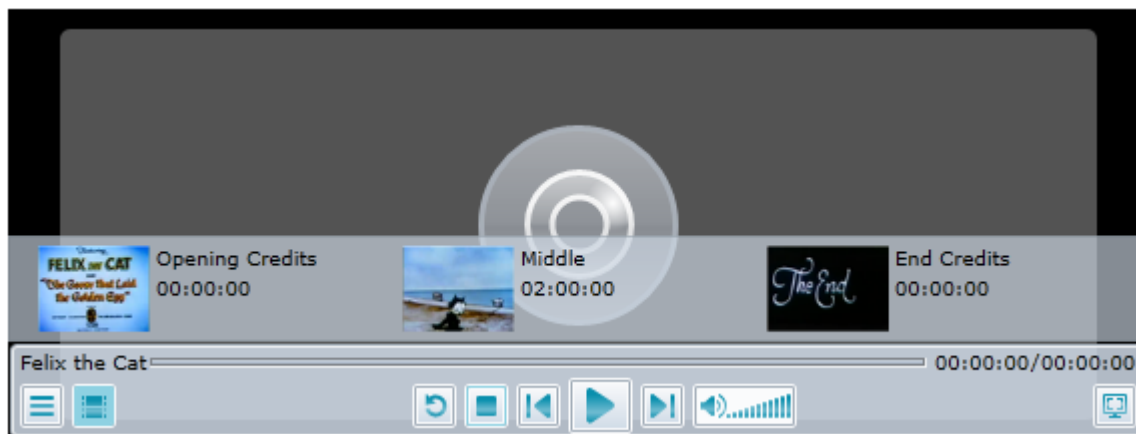
Once your media item is created, you can create chapters for it, which will then be added to a chapter list. For more information on chapter lists, see the [Chapter List](#) (page 23) topic.

### Accessing the Item List

You can access the item list by pressing the **Item List** button  on the C1MediaPlayer control. If you prefer to have the item list opened at startup, you can set the **C1MediaPlayer.IsItemListVisible** property to **True**.

### Chapter List

The chapter list displays all of the chapters added to the current media item. You can use the right and left arrow buttons to scroll through the list of chapters and then click on a chapter to select it. The chapter list looks as follows:



A few properties will have to be set to achieve the look and behavior of a chapter list such as the one above. To title a media item, you have to set the **C1MediaChapter.Title** property to a string. You can add the thumbnail by setting the **C1MediaChapter.ThumbnailSource** property to an image file. Finally, you will need to specify the time that the chapter starts at by setting the **C1MediaChapter.Position** property to a time.

To create a chapter list like the one in the image above, you would use the following XAML:

```
<clmediaplayer:C1MediaChapter
  Title="Opening Credits"
  Position="00:00:00"
  ThumbnailSource="FelixOpen.png"/>
<clmediaplayer:C1MediaChapter
  Title="Middle"
```


```
Position="00:03:15"
ThumbnailSource="FelixMiddle.png"/>
<clmediaplayer:C1MediaChapter
Title="End Credits"
Position="00:07:04"
ThumbnailSource="FelixEnd.png"/>
```

For task-based help about adding chapters to a media item, see [Creating Chapters](#) (page 32).

### Accessing the Chapter List

You can access the item list by pressing the **Chapter List** button  on the C1MediaPlayer control. If you prefer to have the item list opened at startup, you can set the IsChapterListVisible property to **True**.

### Adjust Volume

Users can control the volume of any media by manipulating the adjust volume slider. The adjust volume slider works just like the Windows Media Player slider: sliding it to the left decreases volume, sliding it to the right increases the volume, and clicking the mute button  mutes the sound.

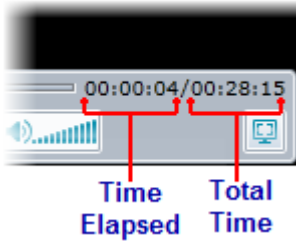


### Controlling the Initial Volume

You can specify the initial volume of the C1MediaPlayer control by setting the Volume property to a value between 0 and 1. A value of 0 turns the volume off, a value of 1 turns the volume up to its maximum, and a value of 0.5 places the volume in the middle of the scale. If you'd rather mute the control at run time, you can also set the IsMuted property to **True**.

### Time Presenter

The position element provides two time markers: time elapsed and total time. The time elapsed is on the left side of the time presenter while the total time is on the right side of the time presenter.



By default, the time elapsed will start from zero. However, you can change the starting position of a media file by setting the Position property to a time. When you run the project, the media file will begin from the point that you specified, and the time elapsed portion of the time presenter will start counting time from that position of time.

## Supported File Types

The C1MediaPlayer control currently supports the following media types:

### Video

The video formats supported are as follows:

- Raw Video.
- YV12 - YCrCb(4:2:0).
- RGBA - 32 bit Alpha Red, Green, Blue.
- WMV, MV2, and MV3 ( Windows Media Video 7, 8, and 9)
- Supports Simple and Main Profiles.
- Supports only progressive (non-interlaced) content.
- WMVA: Windows Media Video Advanced Profile, non-VC-1.
- WVC1: Windows Media Video Advanced Profile, VC-1.
- Supports Advanced Profile.
- Supports only progressive (non-interlaced) content.
- H264 (ITU-T H.264 / ISO MPEG-4 AVC).

### Audio

The audio formats supported are as follows:

- "1" –WAV format.
- "353" - Microsoft Windows Media Audio v7, v8 and v9.x Standard (WMA Standard).
- "354" - Microsoft Windows Media Audio v9.x and v10 Professional (WMA Professional).
- "85" - ISO MPEG-1 Layer III (MP3).
- "255" - ISO Advanced Audio Coding (AAC).

# MediaPlayer for WPF Layout and Appearance

The following topics detail how to customize the C1MediaPlayer 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 layout the control and to customize the control's actions.

## 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.

## C1MediaPlayer ClearStyle Properties

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

The following table outlines the brush properties of the **C1MediaPlayer** control:

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

You can completely change the appearance of the **C1MediaPlayer** control by setting a few properties, such as the **Background** property, which sets the background color of the media player. For example, if you set the **Background** property to "#FFE40005", the **C1MediaPlayer** control would appear similar to the following:



If you want the buttons to appear lime green for better contrast against the red, you can also set the **ButtonForeground** and **ButtonBackground** properties. In the following example, the **ButtonForeground** property is set to “#FF5500DE” and the **ButtonBackground** property is set to “#FF00F500”.



It's that simple with ComponentOne's ClearStyle technology. For more information on ClearStyle, see the [ComponentOne ClearStyle Technology](#) (page 25) topic.

# MediaPlayer for WPF Appearance Properties

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

## Text Properties

The following properties let you customize the appearance of text in the **C1MediaPlayer** control.

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

## Color Properties

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

Property	Description
<a href="#">Background</a>	Gets or sets a brush that describes the background of a control. This is a dependency property.
<a href="#">Foreground</a>	Gets or sets a brush that describes the foreground color. This is a dependency property.

## Border Properties

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

Property	Description
<a href="#">BorderBrush</a>	Gets or sets a brush that describes the border background of a control. This is a dependency property.
<a href="#">BorderThickness</a>	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 **C1MediaPlayer** control.

Property	Description
<a href="#">ActualHeight</a>	Gets or sets the horizontal alignment characteristics applied to this element when it is composed within a

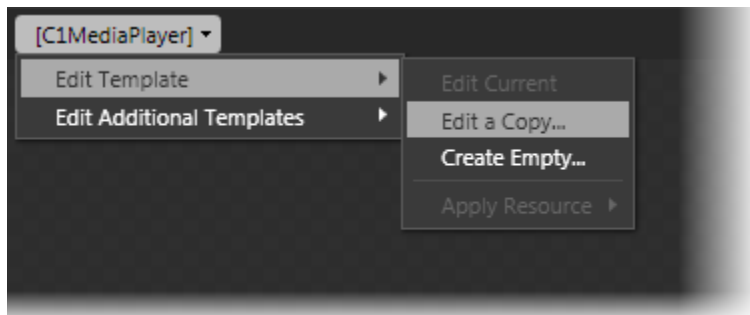
	parent element, such as a panel or items control. This is a dependency property.
<a href="#">ActualWidth</a>	Gets the rendered width of this element. This is a dependency property.
<a href="#">Height</a>	Gets or sets the suggested height of the element. This is a dependency property.
<a href="#">MaxHeight</a>	Gets or sets the maximum height constraint of the element. This is a dependency property.
<a href="#">MaxWidth</a>	Gets or sets the maximum width constraint of the element. This is a dependency property.
<a href="#">MinHeight</a>	Gets or sets the minimum height constraint of the element. This is a dependency property.
<a href="#">MinWidth</a>	Gets or sets the minimum width constraint of the element. This is a dependency property.
<a href="#">Width</a>	Gets or sets the width of the element. This is a dependency property.

## Templates

One of the main advantages to using a WPF control is that controls are "lookless" with a fully customizable user interface. Just as you design your own user interface (UI), or look and feel, for WPF applications, you can provide your own UI for data managed by **ComponentOne MediaPlayer for WPF**. Extensible Application Markup Language (XAML; pronounced "Zammel"), an XML-based declarative language, offers a simple approach to designing your UI without having to write code.

### Accessing Templates

You can access templates in Microsoft Expression Blend by selecting the C1MediaPlayer control and, in the menu, selecting **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or **Create Empty** to create a new blank template.



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

Note that you can use the [Template](#) property to customize the template.

### Included Templates

Additional templates are also available for C1MediaPlayer elements. To access these templates, select the C1MediaPlayer control and, in the menu, select **Edit Additional Templates**. Select **Edit a Copy** to create an editable copy of the current template or select **Create Empty** to create a new blank template.

The additional templates described in the table below.

Template	Description
BufferingTemplate	Template for an overlay over the screen when the state of the content is buffering.
PausedTemplate	Template for an overlay over the screen when the state of the content is paused.
ScreenExtension	Template for an optional overlay of the video area.
ToolBarExtension	Template for an optional extension of the ToolBar that contains the show/hide items/chapter list.
UnstartedTemplate	Template for an overlay over the screen when the state of the content is unstarted.

## MediaPlayer for WPF Samples

Please be advised that these ComponentOne software tools are accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studios.

Samples can be accessed from the **ComponentOne Control Explorer**. To view samples, on your desktop, click the **Start** button and then click **ComponentOne | Studio for WPF | Samples | WPF ControlExplorer**.

The following pages within the **ControlExplorer** detail the C1MediaPlayer control:

Sample	Description
MediaPlayer	Illustrates the functionality of the C1MediaPlayer control.

## MediaPlayer for WPF Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET and know how to use the C1MediaPlayer control in general. If you are unfamiliar with the **ComponentOne MediaPlayer for WPF** product, please see the **MediaPlayer for WPF** Quick Start first.

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


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

### Adding Media Content

You can add media content by creating a C1MediaItem object and then setting its MediaSource property to the location of your media file. In this topic, you will add a video file to the media player in the **Properties** window, in XAML, and in code.

#### At Design Time

To add video content, complete the following steps:

1. Click the C1MediaPlayer control once to select it.
2. Find the **Items** property and click its ellipsis button .

The **Collection Editor: Items** dialog box opens.

3. Click **Add** to add a `C1MediaItem` object to the `C1MediaPlayer` control.
4. In the **Properties** grid, set the following:
  - Set the `MediaSource` property to `"http://download.componentone.com/pub/Videos/Trevor%20Does%20Silverlight.wmv"`.
  - Set the `Title` property to `"Trevor Does Silverlight"`.
5. Click the **OK** to close the **Collection Editor: Items** dialog box.
6. Run the program and observe that content is loaded into the control.

### In XAML

To add video content using XAML, complete the following:

1. Place the following markup between the `<clmediaplayer:C1MediaPlayer>` and `</clmediaplayer:C1MediaPlayer>` tags:

```
<clmediaplayer:C1MediaItem
  MediaSource="http://download.componentone.com/pub/Videos/Trevor%20Does%20
  Silverlight.wmv" Title="Trevor Does Silverlight" Name="C1MediaPlayer1" />
```

2. Run the program and observe that content is loaded into the control.

### In Code

To add video content using code, complete the following steps:

1. Open the `Window1.xaml.cs` page and import the following namespace:

- Visual Basic  
`Imports C1.WPF.MediaPlayer`
- C#  
`using C1.WPF.MediaPlayer;`

2. Place the following code beneath the `InitializeComponent()` method:

- Visual Basic

```
'Create a C1MediaItem object
Dim C1MediaItem1 As New C1MediaItem()

'Create a Uri object that contains the media file's path
Dim Uri1 As New
Uri("http://download.componentone.com/pub/Videos/Trevor%20Does%20Silver
light.wmv")

'Set the C1MediaItems content source to the Uri object
C1MediaItem1.MediaSource = Uri1

'Name the media item
C1MediaItem1.Title = "Trevor Does Silverlight"

'Add the media item to the media player
C1MediaPlayer1.Items.Add(C1MediaItem1)
```

- C#

```
//Create a C1MediaItem object
C1MediaItem C1MediaItem1 = new C1MediaItem();

//Create a Uri object that contains the media file's path
Uri Uri1 = new
Uri("http://download.componentone.com/pub/Videos/Trevor%20Does%20Silver
light.wmv");

//Set the C1MediaItems content source to the Uri object
C1MediaItem1.MediaSource = Uri1;

//Name the media item
C1MediaItem1.Title = "Trevor Does Silverlight";

//Add the media item to the media player
c1MediaPlayer1.Items.Add(C1MediaItem1);
```




3. Run the program and observe that content is loaded into the control.

## Creating Chapters

You can create chapters by creating a `C1MediaChapter` item and then settings its `Position` property. This topic assumes that you have completed the [Adding Media Content](#) (page 30) task-based help topic.

### At Design Time

To create a chapter, complete the following steps:


1. Click the `C1MediaPlayer` control once to select it.
2. In the Properties window, click the **Items** property's ellipsis button . The **Collection Editor: Items** dialog box opens.
3. In the **Items** pane, select the `C1MediaItem` that you want to add the chapters to.
4. In the **Properties** pane, click the **Chapters** property's ellipsis button .
5. The **Collection Editor: Chapters** dialog box opens.
6. Create a chapter by completing the following steps:
  - a. Click **Add** to add a chapter.  
A `C1MediaChapter` item is added to the **Items** pane.
  - b. Set the `Position` property to a time. For this example, set it to "00:01:25", which will create the chapter at 1 minute and 25 seconds into media file.
  - c. Set the `Title` property to a name. For this example, set it "Trevor Looks Confused".
7. Click **OK** to close the **Collection Editor: Chapters** dialog box.
8. Click **OK** to close the **Collection Editor: Items** dialog box.
9. Run the program and click the **Chapter List** button . Observe that one chapter, **Trevor Looks Confused**, appears in the list. You can double-click it to skip ahead to that part of the media file.

### In XAML

To create a chapter, complete the following steps:

1. Place the following markup between the `<clmediaplayer:C1MediaItem>` and the `</clmediaplayer:C1MediaItem>` tags:

```
<clmediaplayer:C1MediaChapter Name="C1MediaPlayer1"
    Position="00:01:25" Title="Trevor Looks Confused"/>
```

2. Run the program and click the **Chapter List** button . Observe that one chapter, **Trevor Looks Confused**, appears in the list. You can double-click it to skip ahead to that part of the media file.

## Turning Off Autoplay

By default, the C1MediaPlayer control will autoplay audio files; however, you can turn this feature off by setting the `AutoPlay` property to **False**.

### At Design Time

To turn off autoplay, complete the following steps:

1. Click the C1MediaPlayer control once to select it.
2. In the **Properties** window, locate the `AutoPlay` check box and deselect it.

### In XAML

To turn off autoplay, add `AutoPlay="False"` to the `<clmediaplayer:C1MediaPlayer>` tag so that the markup resembles the following:

```
<clmediaplayer:C1MediaPlayer Name="C1MediaPlayer1" AutoPlay="False">
```

### In Code

To turn off autoplay, complete the following steps:

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

- Visual Basic  
`C1MediaPlayer1.AutoPlay = False`
- C#  
`c1MediaPlayer1.AutoPlay = false;`

3. Run the program.

## Looping Media Files

By default, the C1MediaPlayer control will automatically run through its item list only once. But if you'd prefer to have the player continuously loop through its items, you can set the `IsLooping` property to **True**.

### At Design Time

To turn on looping, complete the following steps:

1. Click the C1MediaPlayer control once to select it.
2. In the **Properties** window, locate the `IsLooping` check box and select it.

### In XAML

To turn on looping, add `IsLooping="True"` to the `<clmediaplayer:C1MediaPlayer>` tag so that the markup resembles the following:

```
<clmediaplayer:C1MediaPlayer Name="C1MediaPlayer1" IsLooping="True">
```

### In Code

To turn on looping, complete the following steps:

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

- Visual Basic  
`C1MediaPlayer1.IsLooping = True`
- C#  
`c1MediaPlayer1.IsLooping = true;`

3. Run the program.

## Setting the Initial Volume

You can set the initial volume setting of the C1MediaPlayer control by setting the Volume property to a value between 0 and 1. In this topic, you will set the volume so that it's at its midway point upon page load.

### At Design Time

To set the volume, complete the following steps:

1. Click the C1MediaPlayer control once to select it.
2. In the **Properties** window, set the Volume property to "0.5".
3. Run the program and observe that the C1MediaPlayer control's volume is set halfway.

### In XAML

Complete the following steps:

1. To set the volume, complete the following steps:
2. Add `Volume="0.5"` to the `<clmediaplayer:C1MediaPlayer>` tag so that the markup resembles the following:

- 3.

```
<clmediaplayer:C1MediaPlayer Name="C1MediaPlayer1" Volume="0.5">
```

4. Run the program and observe that the C1MediaPlayer control's volume is set halfway.

### In Code

To set the volume, complete the following steps:

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

- Visual Basic  
`C1MediaPlayer1.Volume = 0.5`
- C#  
`c1MediaPlayer1.Volume = 0.5;`

3. Run the program and observe that the C1MediaPlayer control's volume is set halfway.

## Showing the Chapter List on Page Load

By default, the C1MediaPlayer control doesn't show the chapter list when it's loaded; users will have to open the list themselves by clicking the **Chapter List** button. However, you can make the chapter list show upon page load by setting the IsChapterListVisible property to **True**.

### At Design Time

To open the chapter list on page load, complete the following steps:

1. Click the C1MediaPlayer control once to select it.
2. In the **Properties** window, locate the IsChapterListVisible check box and then select it.
3. Run the program and observe that the chapter list opens at run time.

### In XAML

To open the chapter list on page load, complete the following steps:

1. Add `IsChapterListVisible="True"` to the `<clmediaplayer:C1MediaPlayer>` tag so that the markup resembles the following:

```
<clmediaplayer:C1MediaPlayer Name="C1MediaPlayer1"
    IsChapterListVisible="True">
```

2. Run the program and observe that the chapter list opens at run time.

### In Code

To open the chapter list on page load, complete the following steps:

1. Open the Window1.xaml.cs page.
2. Place the following code beneath the **InitializeComponent()** method:
  - Visual Basic  
`C1MediaPlayer1.IsChapterListVisible = True`
  - C#  
`c1MediaPlayer1.IsChapterListVisible = true;`
3. Run the program and observe that the chapter list opens at run time.

## Showing the Item List on Page Load

By default, the C1MediaPlayer control doesn't show the item list when it's loaded; users will have to open the list themselves by clicking the **Item List** button. However, you can make the item list show upon page load by setting the IsItemListVisible property to **True**.

### At Design Time

To open the item list on page load, complete the following steps:

1. Click the C1MediaPlayer control once to select it.
2. In the **Properties** window, locate the IsItemListVisible check box and then select it.
3. Run the program and observe that the item list opens at run time.

### In XAML

To open the item list on page load, complete the following steps:

1. Add `IsItemListVisible="True"` to the `<clmediaplayer:C1MediaPlayer>` tag so that the markup resembles the following:

```
<clmediaplayer:C1MediaPlayer Name="C1MediaPlayer1"  
IsItemListVisible="True">
```

2. Run the program and observe that the item list opens at run time.

### In Code

To open the item list on page load, complete the following steps:

1. Open the `Window1.xaml.cs` page.
2. Place the following code beneath the `InitializeComponent()` method:
  - Visual Basic  
`C1MediaPlayer1.IsItemListVisible = True`
  - C#  
`c1MediaPlayer1.IsItemListVisible = true;`
3. Run the program and observe that the item list opens at run time.