
ComponentOne

DockControl for WPF

Copyright © 2012 ComponentOne LLC. All rights reserved.

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue
3rd Floor
Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com

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

Sales

E-mail: sales@componentone.com

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

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 DockControl for WPF | 1 |
| Installing DockControl for WPF | 1 |
| Studio for WPF Setup Files | 1 |
| Using Maps Powered by Esri | 2 |
| System Requirements | 3 |
| Installing Demonstration Versions | 4 |
| Uninstalling DockControl for WPF | 4 |
| End-User License Agreement | 4 |
| Licensing FAQs | 4 |
| What is Licensing? | 4 |
| How does Licensing Work? | 5 |
| Common Scenarios | 5 |
| Troubleshooting | 7 |
| Technical Support | 9 |
| Redistributable Files | 10 |
| 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 DockControl for WPF Components to a Blend Project | 14 |
| Adding the DockControl for WPF Components to a Visual Studio Project | 14 |
| DockControl for WPF Features | 17 |
| DockControl for WPF Quick Start | 17 |
| Step 1 of 3: Creating a WPF Application | 17 |
| Step 2 of 3: Adding a C1DockTabControl with C1DockTabItems | 18 |
| Step 3 of 3: Running the Application | 19 |
| XAML Quick Reference | 21 |
| Working with DockControl for WPF | 22 |
| C1DockControl Elements | 22 |

| | |
|---|----|
| Docking Options..... | 23 |
| Docking Diamond and Zones..... | 24 |
| DockControl for WPF Layout and Appearance | 26 |
| Templates | 26 |
| ComponentOne ClearStyle Technology | 26 |
| How ClearStyle Works | 27 |
| DockControl ClearStyle Properties..... | 27 |
| C1DockControl Themes | 28 |
| DockControl for WPF Samples..... | 31 |
| DockControl for WPF Task-Based Help..... | 31 |
| Setting the Dock Mode..... | 31 |

ComponentOne DockControl for WPF

Handle multiple windows in your WPF application with **ComponentOne DockControl™ for WPF**. Similar to the docking system in Microsoft Visual Studio, **C1DockControl** delivers dockable, floating, and tabbed windows. You can also auto-hide sections and easily style the **C1DockControl**.

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



Getting Started

- [Quick Start](#) (page 17)
- [Working with DockControl for WPF](#) (page 22)
- [Task-Based Help](#) (page 31)

Installing DockControl for WPF

The following sections provide helpful information on installing **ComponentOne DockControl 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 DockControl for WPF**, the following DLLs are installed:
- C1.WPF.dll
 - C1.WPF.Docking.dll
 - C1.WPF.Docking.Expression.Design.4.0.dll
 - C1.WPF.Docking.Expression.Design.dll
 - C1.WPF.Docking.VisualStudio.Design.4.0.dll
 - C1.WPF.Docking.VisualStudio.Design.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.

- C1WPF\XAML** Contains the full XAML definitions of C1DockControl styles and templates which can be used for creating your own custom styles and templates.

The **ComponentOne Studio for WPF Help Setup** program installs integrated Microsoft Help 2.0 and Microsoft Help Viewer help to the C:\Program Files\ComponentOne\Studio for WPF directory in the following folders:

| | |
|-------------------|---|
| H2Help | Contains Microsoft Help 2.0 integrated documentation for all Studio components. |
| HelpViewer | Contains Microsoft Help Viewer Visual Studio 2010 integrated documentation for all Studio components. |

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:

| | |
|---------------|---|
| Common | Contains support and data files that are used by many of the demo programs. |
| C1WPF | Contains samples for DockControl for WPF . |

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

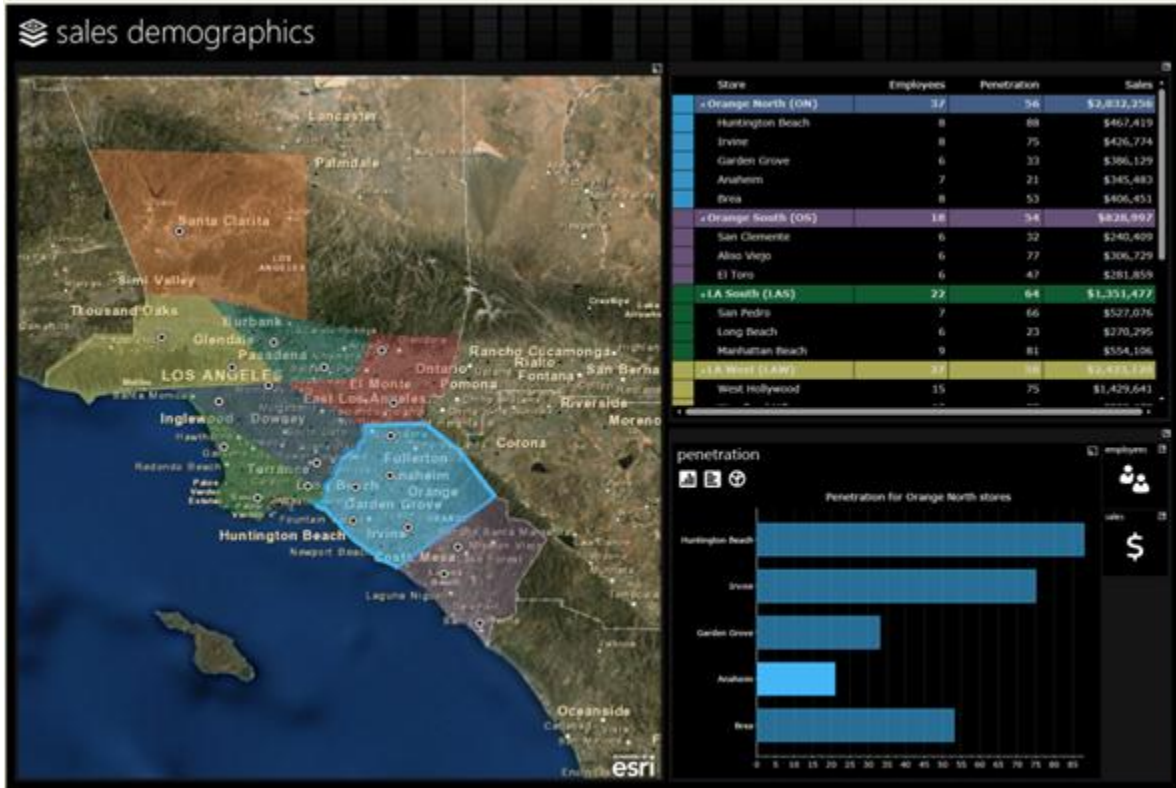
Using Maps Powered by Esri

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.



Esri 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®** DockControl for WPF includes design-time support for

Blend Compatibility: Expression Blend.

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

Installing Demonstration Versions

If you wish to try **ComponentOne DockControl 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 DockControl for WPF

To uninstall **ComponentOne DockControl 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.

To uninstall **ComponentOne DockControl for WPF** integrated help:

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

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 project and go to the Solution Explorer window.
2. Click the **Show All Files** button on the top of the window.
3. Find the **licenses.licx** file and open it. If prompted, continue to open the file.
4. Change the version number of each component to the appropriate value. If the component does not appear in the file, obtain the appropriate data from another **licenses.licx** file or follow the alternate procedure following.
5. Save the file, then close the **licenses.licx** tab.
6. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

Alternatively, follow these steps:

1. Open the project and go to the Solution Explorer window.
2. Click the **Show All Files** button on the top of the window.
3. Find the **licenses.licx** file and delete it.
4. Close the project and reopen it.
5. Open the main form and add an instance of each licensed control.
6. Check the Solution Explorer window, there should be a **licenses.licx** file there.
7. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

For ASP.NET 2.x applications, follow these steps:

1. Open the project and go to the Solution Explorer window.
2. Find the **licenses.licx** file and right-click it.
3. Select the **Rebuild Licenses** option (this will rebuild the **App_Licenses.licx** file).
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**).

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.
- **Product Forums**
ComponentOne's [product forums](#) are available for users to share information, tips, and techniques regarding ComponentOne products. ComponentOne developers will be available on the forums to share insider tips and technique and answer users' questions. 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 DockControl 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.Docking.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 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 |
|--|--|
| <code>xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"</code> | This is the default Windows Presentation Foundation namespace. |
| <code>xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"</code> | This is a XAML namespace that is mapped to the x: prefix. The x: prefix provides a quick, easy way to reference the namespace, which defines many commonly-used features necessary for WPF applications. |

When you add a `C1DockControl` 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:

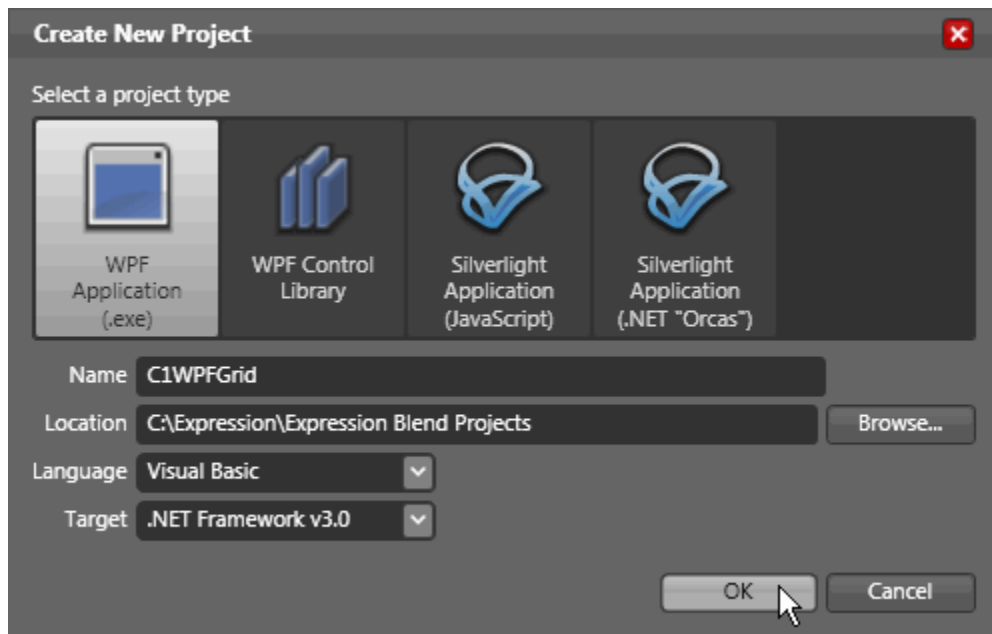
```
xmlns:MyMTB=http://schemas.componentone.com/wpf/C1Docking
```

You can now use your custom namespace when assigning properties, methods, and events.

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**.

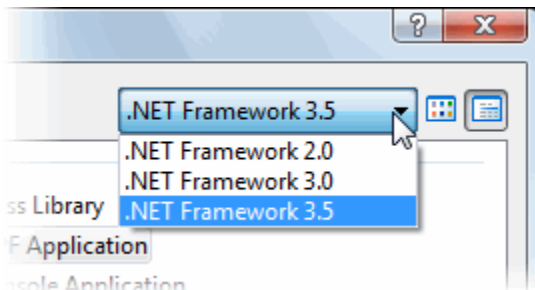


A new Blend project with a XAML window is created.

Creating a .NET Project in Visual Studio

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

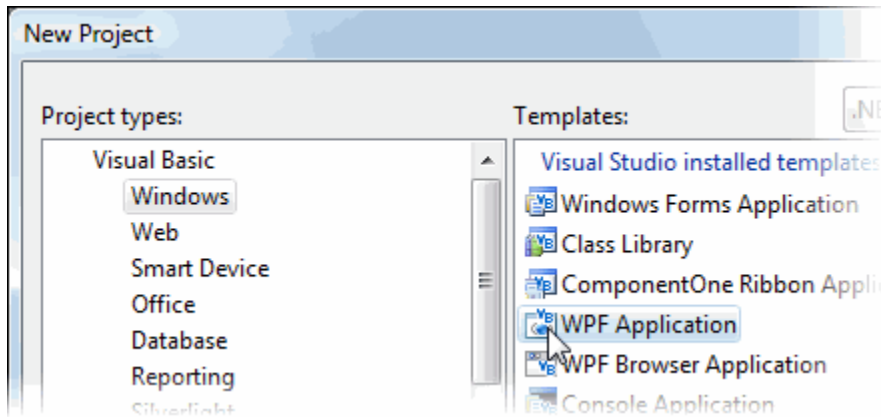
1. From the **File** menu in Microsoft Visual Studio 2008, select **New Project**.
The **New Project** dialog box opens.
2. Choose the appropriate .NET Framework from the Framework drop-down box in the top-right of the dialog box.



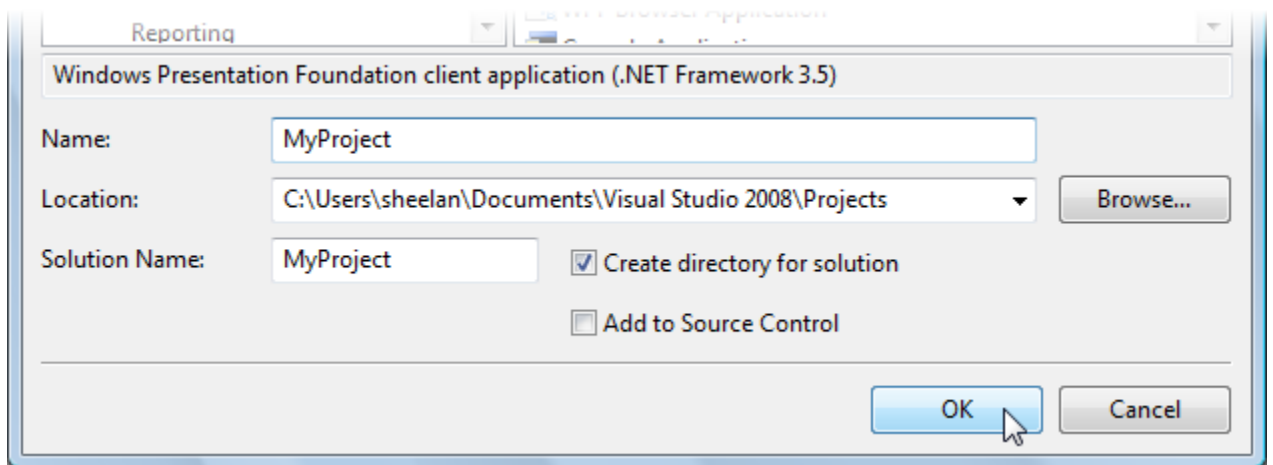
3. Under **Project types**, select either **Visual Basic** or **Visual C#**.

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**.



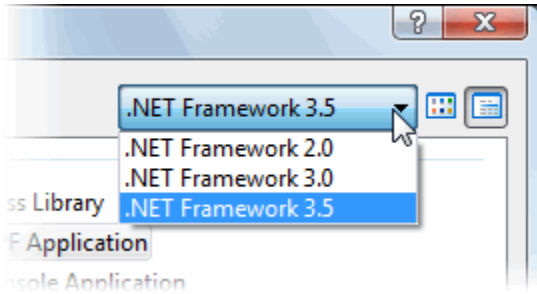
A new Microsoft Visual Studio .NET WPF project is created with a XAML file that will be used to define your user interface and commands in the application.

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 DockControl for WPF Components to a Blend Project

In order to use **C1DockControl** or another **ComponentOne DockControl for WPF** element 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.Docking.dll** assembly installed with **DockControl for WPF**.

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

3. Select **C1.WPF.Docking.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.Docking** assembly, click the **Asset Library** button  in the Blend Toolbox. The **Asset Library** appears:
2. Click the **Controls** drop-down arrow and select **All**.
3. Select **C1DockControl**. The component will appear in the Toolbox below the **Asset Library** button.
4. Double-click the **C1DockControl** component in the Toolbox to add it to **Window1.xaml**.

Adding the DockControl for WPF Components to a Visual Studio Project

When you install **ComponentOne DockControl for WPF** the **C1DockControl** control should be added to your Visual Studio Toolbox. You can also manually add ComponentOne controls to the Toolbox.

ComponentOne DockControl for WPF provides the following controls and elements:

- **C1DockControl**

- C1DockGroup
- C1DockTabControl
- C1DockTabItem

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

Manually Adding DockControl for WPF to the Toolbox

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

- C1DockControl
- C1DockGroup
- C1DockTabControl
- C1DockTabItem

To manually add the C1DockControl 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 **DockControl for WPF** components appear on its own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **C1WPFDockControl**, 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.Docking** namespace. Note that there may be more than one component for each namespace.

Adding DockControl for WPF to the Window

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

1. Add the C1DockControl control to the Visual Studio Toolbox.
2. Double-click C1DockControl or drag the control onto the window.

Adding a Reference to the Assembly

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

1. Select the **Add Reference** option from the **Project** menu of your project.
2. Select the **ComponentOne DockControl for WPF** assembly from the list on the **.NET** tab or on the **Browse** tab, browse to find the **C1.WPF.Docking.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.Docking
```

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

DockControl for WPF Features

The following are some of the main features of C1DockControl that you may find useful:

- **Docking Diamond**

By default, **DockControl** uses Microsoft docking diamonds like Visual Studio. See [Docking Diamond and Zones](#) (page 24) for more information.
- **Natural Docking**

Customize **DockControl** to provide natural docking allowing you to dock and float multiple windows within your application. Dock indicators appear when you drag windows over dock zones signaling to dock.
- **Floating Windows**

Allow a window to float above your application in separate windows.
- **Tabbed Windows**

Documents opened in instances of the editor are automatically arranged on tabbed panes.
- **Auto Hide**

See more of your code at one time with auto hide using the pushpin. This allows you to minimize not-in-use tool windows along the edges of the IDE.
- **Rich Programmatic API**

For complete flexibility and to restrict behavior, the docking state of any dockable window can be controlled programmatically through the rich programmatic API.
- **Support for Two Monitors**

You can choose which monitor to display windows on by easily dragging windows from one monitor to the other. This option is only available on the WPF platform.
- **Supports ClearStyle Technology**

DockControl for WPF supports ComponentOne's ClearStyle technology, which allows you to easily change control colors without having to change control templates. By setting a few color properties, you can quickly style the **C1DockControl** elements. See [DockControl ClearStyle Properties](#) (page 27) for more information on the ComponentOne ClearStyle technology.

DockControl for WPF Quick Start

The following quick start guide is intended to get you up and running with **DockControl for WPF**. In this quick start, you'll start in Visual Studio to create a new project, add a C1DockControl to your application, and then add a C1DockTabControl with C1DockTabItems.

Step 1 of 3: Creating a WPF Application

In this step you'll create a WPF application in Visual Studio using **ComponentOne DockControl for WPF**.

To set up your project and add a C1DockControl control to your application, complete the following steps:

1. In Visual Studio, select **File | New | Project**. Note that Visual Studio 2010 is used to complete the steps in this quick start.

2. In the **New Project** dialog box, select a language in the left pane (in this example, C# is used), and in the templates list select **WPF Application**.
3. In the .NET Framework drop-down list, select **.NET Framework 3.5** or later.
4. Enter a **Name** for your project and click **OK**. The **New WPF Application** dialog box will appear.
5. Uncheck the **Host the WPF application in a new Web site** box, if necessary, and click **OK**. The **MainPage.xaml** file opens.
6. In the XAML window of the project, place the cursor between the `<Grid>` and `</Grid>` tags and click once.
7. Navigate to the Toolbox and drag the C1DockControl icon in between the `<Grid>` tags to add the C1DockControl to **Window.xaml**. The XAML markup will now look similar to the following:

```
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
    <Grid x:Name="LayoutRoot">
<c1:C1DockControl></c1:C1DockControl>
    </Grid>
</Window>
```

In the next step, you will add a C1DockTabControl with C1DockTabItems.

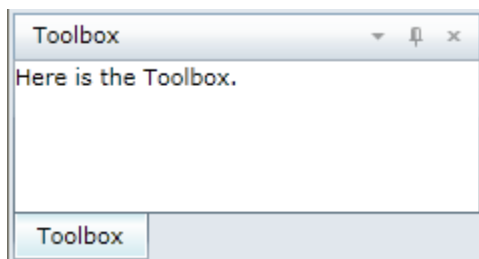
Step 2 of 3: Adding a C1DockTabControl with C1DockTabItems

Next we are going to add a C1DockTabControl with C1DockTabItems.

1. In the XAML markup, place your cursor between the `<c1:C1DockControl></c1:C1DockControl>` tags and press ENTER.
2. Add a C1DockTabControl within these tags using the following XAML markup:

```
<c1:C1DockTabControl Dock="Left">
    <c1:C1DockTabItem Header="Toolbox">
        <TextBlock Text="Here is your Toolbox." />
    </c1:C1DockTabItem>
</c1:C1DockTabControl>
```

This XAML also includes a C1DockTabItem labeled **Toolbox**. If you run the application now, the C1DockTabControl will look similar to the following:



Let's add another C1DockTabItem and C1DockTabControl to show how you can select tabs and dock or float the C1DockTabControl when you run the application.

3. Add XAML for another C1DockTabItem and a C1DockTabControl after the closing </c1:C1DockTabItem> tag, so the full XAML for C1DockControl will look like the following:

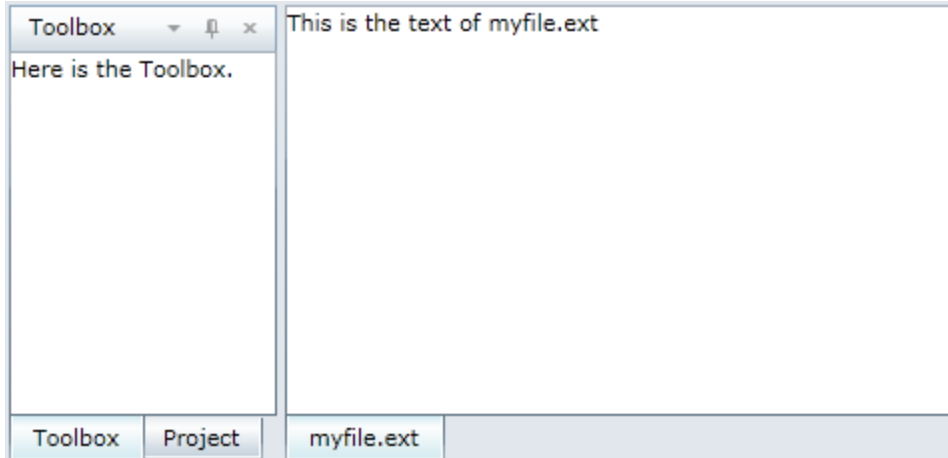
```
<c1:C1DockControl>
  <c1:C1DockTabControl Dock="Left">
    <c1:C1DockTabItem Header="Toolbox">
      <TextBlock Text="Here is the Toolbox." />
    </c1:C1DockTabItem>
    <c1:C1DockTabItem Header="Project">
      <TextBlock Text="Tree of files" />
    </c1:C1DockTabItem>
  </c1:C1DockTabControl>
  <c1:C1DockTabControl DockWidth="500" ShowHeader="False">
    <c1:C1DockTabItem Header="myfile.ext">
      <TextBlock Text="This is the text of myfile.ext" />
    </c1:C1DockTabItem>
  </c1:C1DockTabControl>
</c1:C1DockControl>
```

In the next step you will run the application and change the docking mode for windows.

Step 3 of 3: Running the Application

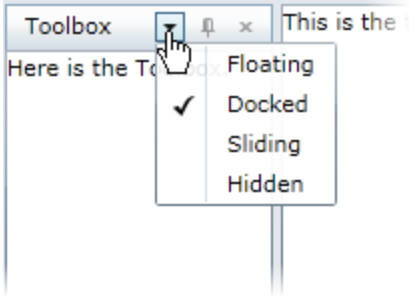
Now that you've created a WPF application with a C1DockControl and tab items, you're ready to run the application. Complete the following steps:

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



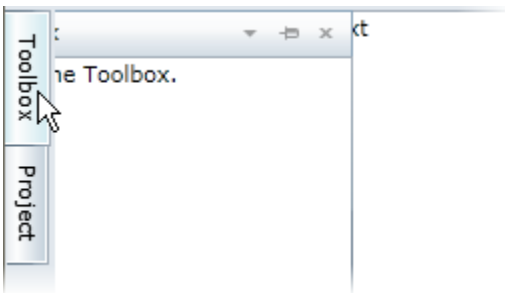
The **Toolbox** is docked on the left as we specified in the XAML.


2. Click the drop-down arrow in the **Toolbox** header and select an option to float, slide, or hide the C1DockControl.



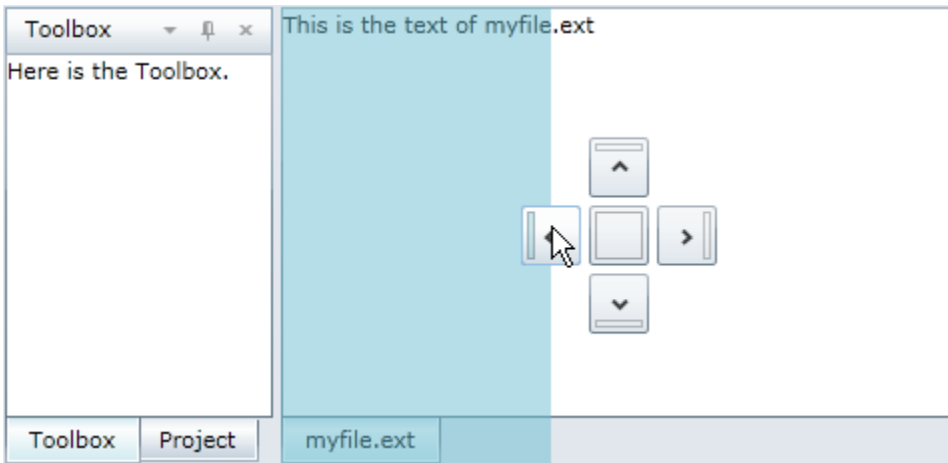
Note that you can also click the **Project** tab and select any of these options for the C1DockControl.

If you click the pushpin , the **C1DockTabItems** will appear on the left and slide open when clicked.



If you click the **Hide** button , the C1DockControl will be hidden.

3. Select the **Toolbox** header and drag it over the C1DockControl with the myfile.txt tab. Notice the dock indicators that appear over the dock zones. The dock zones are shaded in blue when you hover over an indicator.



The second C1DockControl with the myfile.txt C1DockTabItem does not have a header since we set **ShowHeader** to **False**.

Congratulations! You have successfully completed the **DockControl for WPF** quick start. In this quick start, you've used the `C1DockControl` to create several windows that can float or be docked.

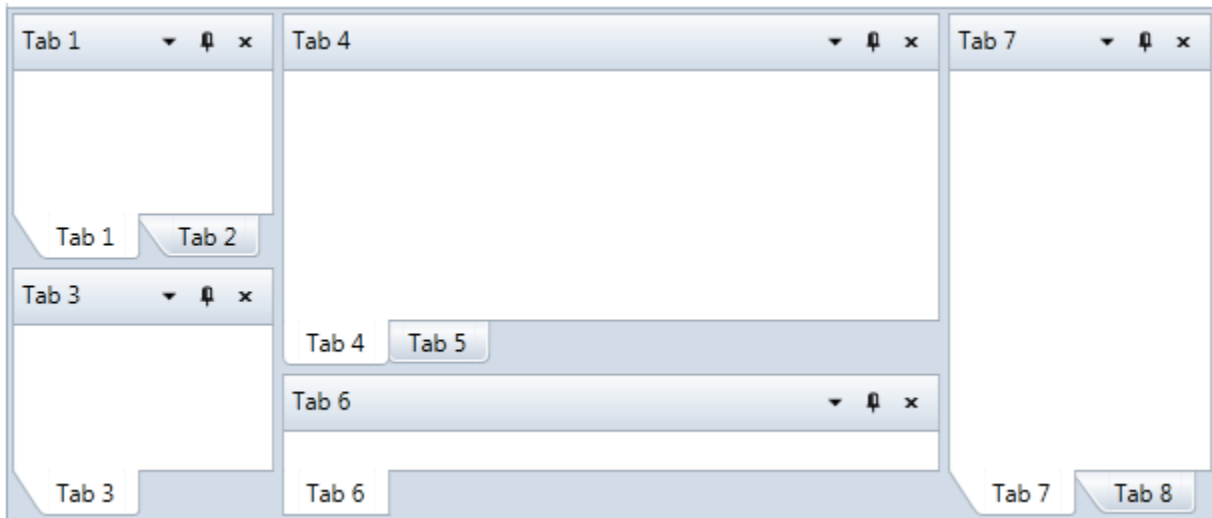
XAML Quick Reference

This topic is dedicated to providing a quick overview of the XAML used to create a `C1DockControl`, `C1DockGroup`, `C1DockTabControl`, and `C1DockTabItem`.

To get started developing, add a `c1` namespace declaration in the root element tag:

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

Here is a sample docking layout:



Below is the XAML for the sample docking layout:

```
<c1:C1DockControl x:Name="dockControl" Margin="-128,0,12,127">
  <c1:C1DockGroup>
    <c1:C1DockTabControl Dock="Top">
      <c1:C1DockTabItem Header="Tab 1" TabShape="Sloped">
        <!--Your Content Here-->
      </c1:C1DockTabItem>
      <c1:C1DockTabItem Header="Tab 2" TabShape="Sloped">
        <!--Your Content Here-->
      </c1:C1DockTabItem>
    </c1:C1DockTabControl>
    <c1:C1DockTabControl>
      <c1:C1DockTabItem Header="Tab 3" TabShape="Sloped"
>
        <!--Your Content Here-->
      </c1:C1DockTabItem>
    </c1:C1DockTabControl>
  </c1:C1DockGroup>
  <c1:C1DockGroup>
    <c1:C1DockTabControl Dock="Top" DockWidth="500"
DockHeight="500" TabItemShape="Rounded">
      <c1:C1DockTabItem Header="Tab 4">
        <!--Your Content Here-->
      </c1:C1DockTabItem>
```

```

        <c1:C1DockTabItem Header="Tab 5">
            <!--Your Content Here-->
        </c1:C1DockTabItem>
    </c1:C1DockTabControl>
    <c1:C1DockTabControl>
        <c1:C1DockTabItem Header="Tab 6">
            <!--Your Content Here-->
        </c1:C1DockTabItem>
    </c1:C1DockTabControl>
</c1:C1DockGroup>
<c1:C1DockTabControl>
    <c1:C1DockTabItem Header="Tab 7" TabShape="Sloped">
        <!--Your Content Here-->
    </c1:C1DockTabItem>
    <c1:C1DockTabItem Header="Tab 8" TabShape="Sloped">
        <!--Your Content Here-->
    </c1:C1DockTabItem>
</c1:C1DockTabControl>
</c1:C1DockControl>

```

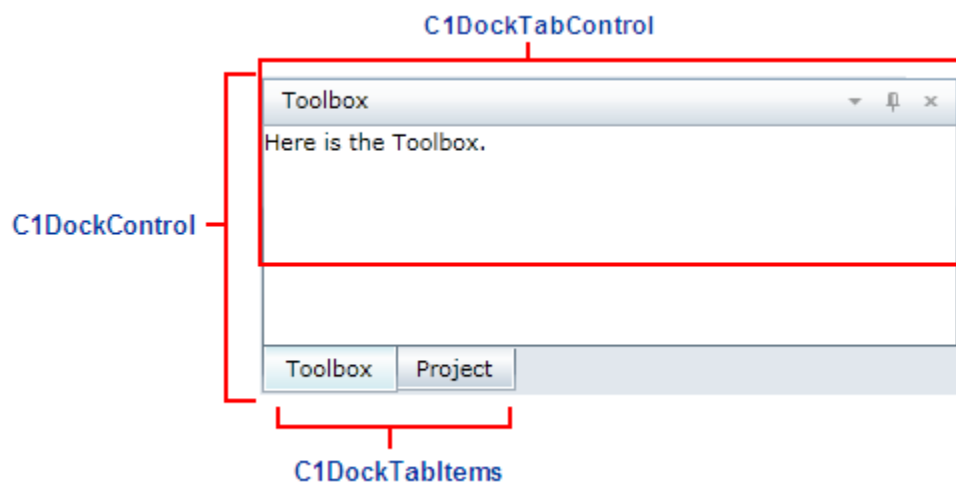
Working with DockControl for WPF

ComponentOne DockControl for WPF includes the `C1DockControl` control, a simple control that allows you to dock, float, or tab windows. The following topics provide more information on the `C1DockControl` and the elements that can be added to it, options for docking windows, and docking indicators and zones.

C1DockControl Elements

The `C1DockControl` serves as a container in which you can add other **DockControl for WPF** elements to help organize your windows.

Once you have a `C1DockControl` on the page, you can add a `C1DockTabControl` with `C1DockTabItems`.



The `C1DockTabControl` contains three buttons:

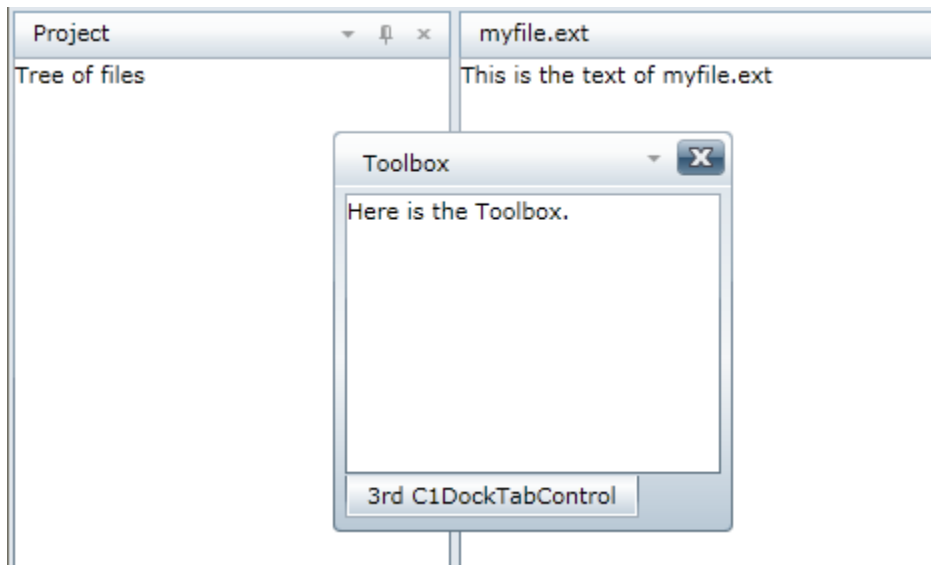
| | |
|-----------------------|--|
| Drop-down list | DockMode options include: Floating, Docked, Sliding, Hidden |
| Pushpin | Enables the auto hide feature. The C1DockTabItems are minimized along the edges of the IDE. |
| Hide | Hides the C1DockTabControl . |

Docking Options

DockControl for WPF provides four DockMode options: Floating, Docked, Sliding, and Hidden. The following images show each of the docking options available in the drop-down list of the C1DockTabControl.

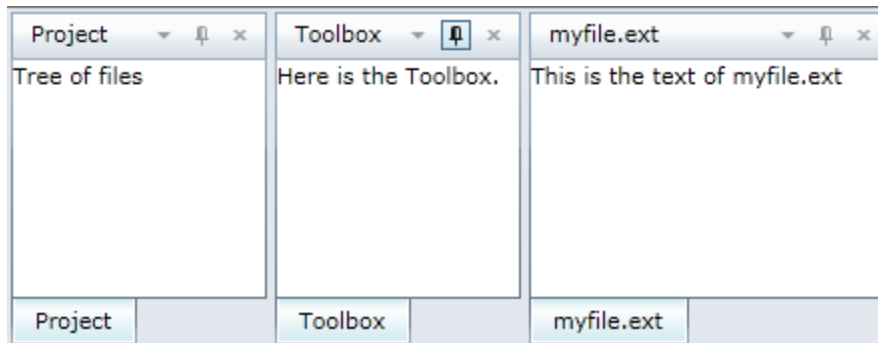
Floating

Click **Floating** to undock the window and allow it to float over the other windows.



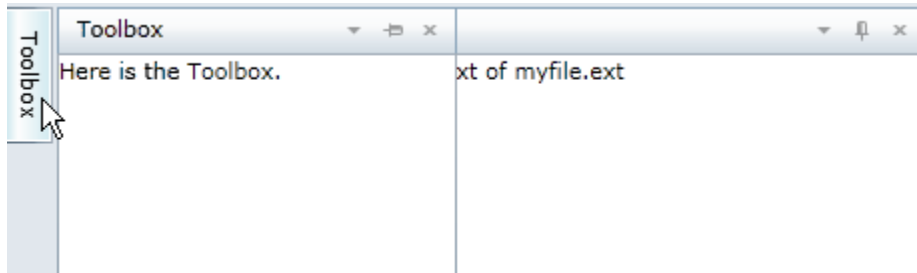
Docked

Click **Docked** to dock the window with the other windows.



Sliding

Click **Sliding** to minimize the C1DockTabItem along the edges of the IDE.



Hidden

Click the **Hide** button to hide the **C1DockTabControl**.

To set the Dock mode:

You can set the dock mode at design time using the `DockMode` property. The following XAML markup sets the dock mode to **Floating**:

```
<c1:C1DockTabControl DockMode="Floating">
  <c1:C1DockTabItem Header="Toolbox">
    <TextBlock Text="Toolbox" />
  </c1:C1DockTabItem>
</c1:C1DockTabControl>
```

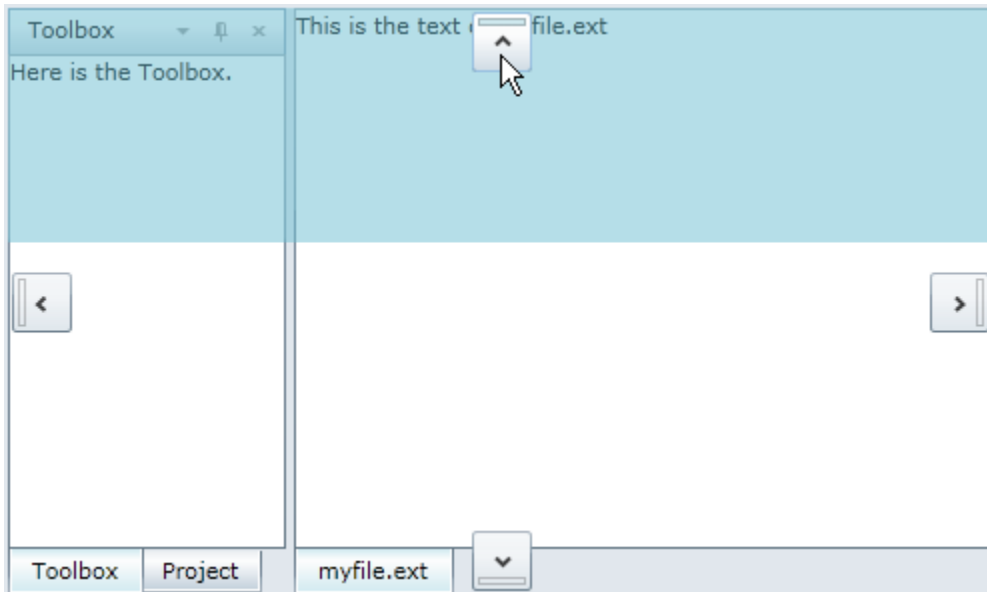
See [Setting the Dock Mode](#) (page 31) for more information.

Docking Diamond and Zones

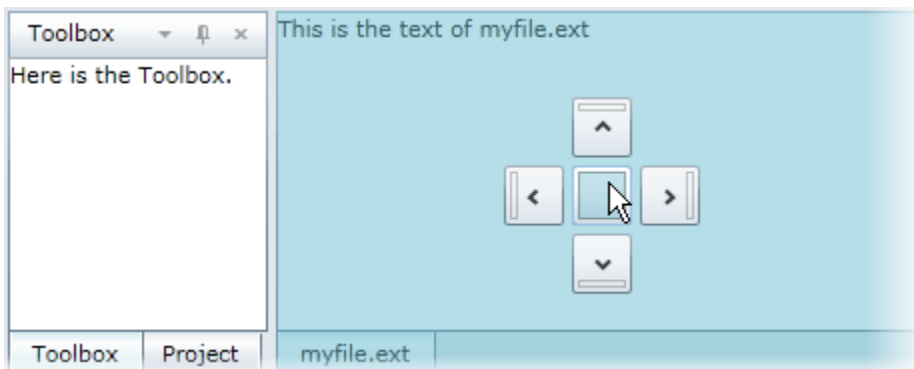
The docking diamond is used to show where `C1DockControl` elements can be docked.

The `C1DockControl` provides four docking indicators so you can dock at the top, right, bottom, or left side of the control.

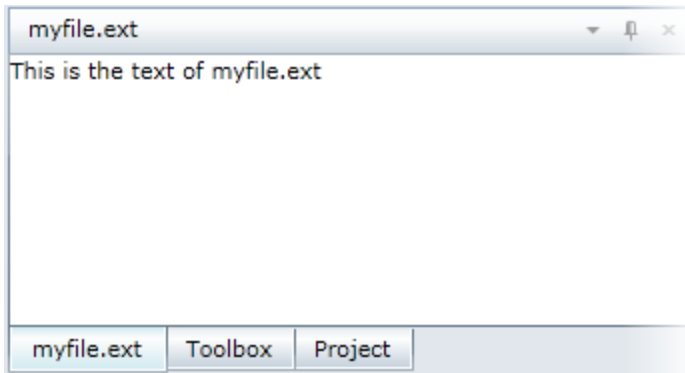
Drag the `C1DockTabControl` header, and blue docking zones will appear showing you where you can dock the window.



If you have multiple **C1DockTabControl**s, drag one of the **C1DockTabControl** headers over one of the other **C1DockTabControl**s. You will notice the docking diamond shows five docking indicators, including top, right, bottom, left, and a center docking indicator that allows you to merge the **C1DockTabItems** into one **C1DockTabControl**.



Using the previous image's example, if the mouse is released over the center indicator, the *myfile.ext* **C1DockTabItem** will appear with the other **C1DockTabItems** in the first **C1DockTabControl**.



DockControl for WPF Layout and Appearance

The following topics detail how to customize the **C1DockControl's** layout and appearance. You can use templates to format and layout the control and to customize the control's actions.

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 DockControl 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 C1DockControl and, in the **Object** menu, selecting **Edit Template**. Select **Edit a Copy** to create an editable copy of the current template or select **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.

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 Silverlight 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.

DockControl ClearStyle Properties

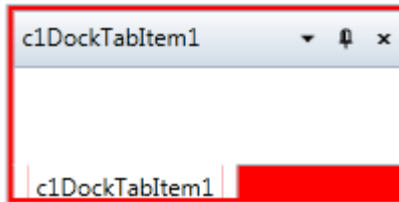
DockControl for WPF supports ComponentOne's ClearStyle technology, which allows you to easily change control colors without having to change control templates. By setting a few color properties, you can quickly style the **C1DockControl** elements. The supported properties for **C1DockControl** are listed in the following table:

| Property | Description |
|----------------------|---|
| Background | Gets or sets the background used to fill the C1DockControl . |
| MouseOverBrush | Gets or sets the brush used to highlight the control when the mouse is hovering over it. |
| TabControlBackground | Gets or sets the background color of the C1DockTabControl . |
| TabControlForeground | Gets or sets the foreground color, or the color of the text, in the C1DockTabControl . |
| TabStripBackground | Gets or sets the background color of the C1DockTabItem . |
| TabStripForeground | Gets or sets the foreground color, or the color of the text, in the C1DockTabItem . |

You can completely change the appearance of the **C1DockControl** by setting these properties. For example, if you set the **C1DockControl.Background** property to **Red** so the XAML markup appears similar to the following:

```
<my:C1DockControl Height="100" Background="Red"
HorizontalAlignment="Left" Margin="176,100,0,0" Name="c1DockControl1"
VerticalAlignment="Top" Width="200">
    <my:C1DockTabControl Height="100"
HorizontalAlignment="Left" Name="c1DockTabControl1"
VerticalAlignment="Top" Width="200">
        <my:C1DockTabItem Header="C1DockTabItem1"
HorizontalAlignment="Left" Name="c1DockTabItem1"
VerticalAlignment="Top" />
    </my:C1DockTabControl>
</my:C1DockControl>
```

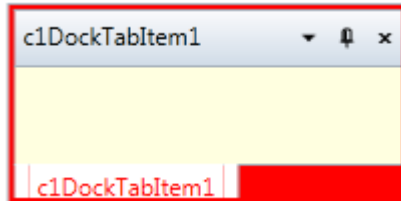
The **C1DockControl** will look similar to the following:



Experiment with the other properties to quickly change the look of the **C1DockControl** elements. For example, the following XAML sets the **Background**, **TabStripForeground**, and **TabControlBackground** properties:

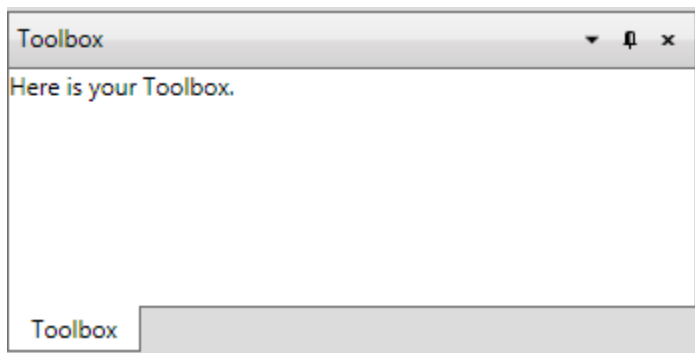
```
<my:C1DockControl Height="100" Background="Red"
  TabStripForeground="Red" TabControlBackground="LightYellow"
  HorizontalAlignment="Left" Margin="176,100,0,0" Name="c1DockControl1"
  VerticalAlignment="Top" Width="200">
  <my:C1DockTabControl Height="100"
    HorizontalAlignment="Left" Name="c1DockTabControl1"
    VerticalAlignment="Top" Width="200">
    <my:C1DockTabItem Header="c1DockTabItem1"
      HorizontalAlignment="Left" Name="c1DockTabItem1"
      VerticalAlignment="Top" />
  </my:C1DockTabControl>
</my:C1DockControl>
```

The **C1DockControl** will look similar to the following:

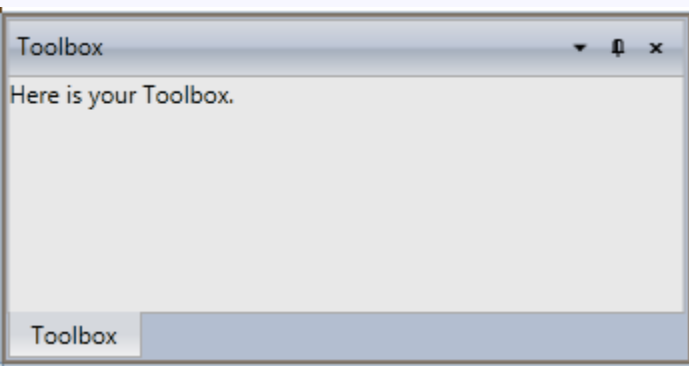
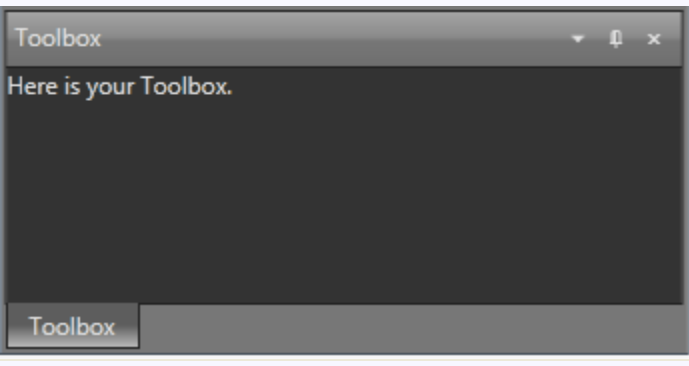
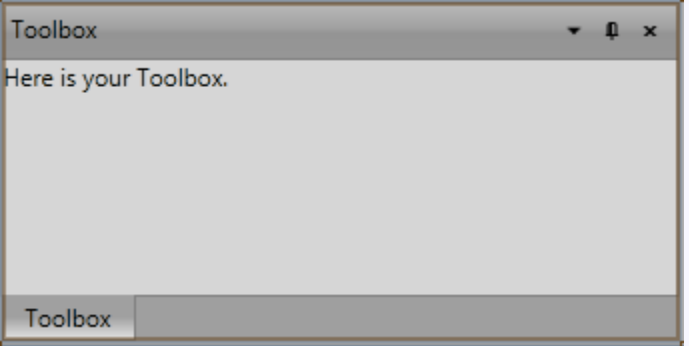
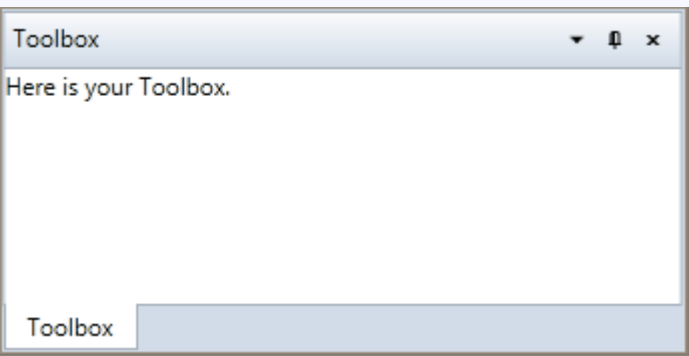


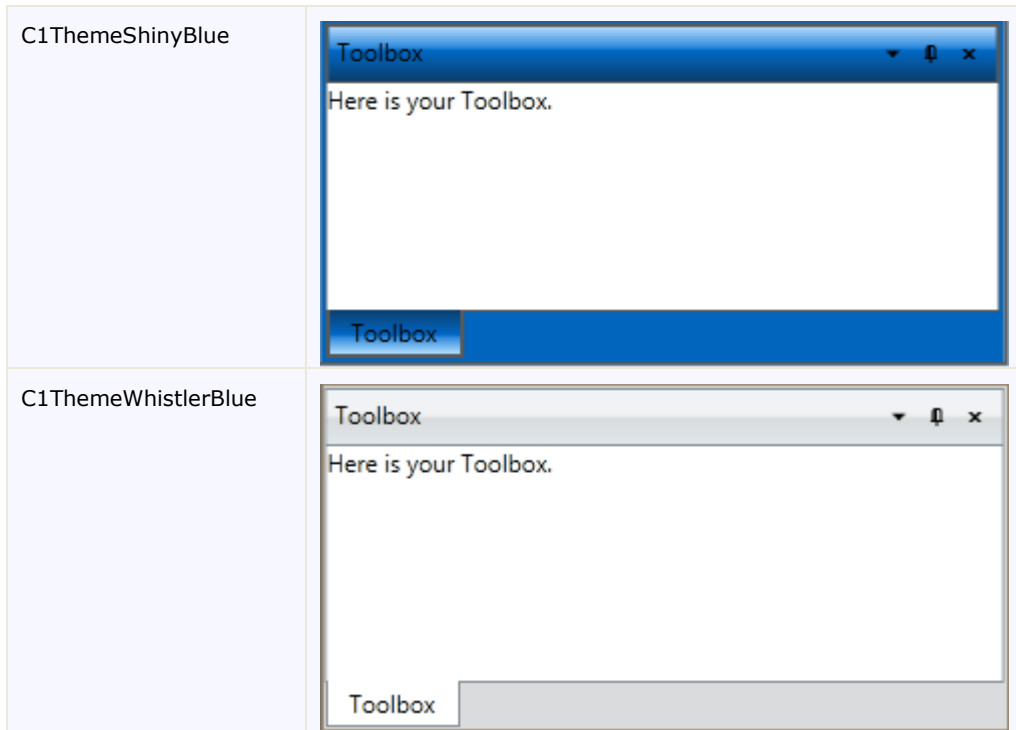
C1DockControl Themes

ComponentOne DockControl for WPF incorporates several themes that allow you to customize the appearance of your grid. When you first add a **C1DockControl** control with a **C1DockTabControl** and **C1DockTabItem(s)** to the page, it appears similar to the following image:



This is the control's default appearance. You can change this appearance by using one of the built-in themes or by creating your own custom theme. All of the built-in themes are based on WPF Toolkit themes. The built-in themes are described and pictured below; note that in the images below, a row has been selected to show selected styles:

| Theme Name | Theme Preview |
|------------------------|--|
| C1ThemeBureauBlack |  |
| C1ThemeExpressionDark |  |
| C1ThemeExpressionLight |  |
| C1Blue |  |



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

- Visual Basic

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

    ' Using ApplyTheme
    C1Theme.ApplyTheme(LayoutRoot, theme)
```

- C#

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

    //Using ApplyTheme
    C1Theme.ApplyTheme(LayoutRoot, theme);
}
```

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

- Visual Basic

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

    ' Using Merged Dictionaries
    Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThem
eResources(theme))
```

End Sub

- C#

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

    //Using Merged Dictionaries
    Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme));
}
```

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

DockControl for WPF Samples

Please be advised that this ComponentOne software tool is 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 **All Programs | ComponentOne | Studio for WPF | Samples | WPF ControlExplorer**.

The following pages within the **ControlExplorer** sample installed with **ComponentOne Studio for WPF** detail the C1DockControl's functionality:

C# Sample

| Sample | Description |
|----------------|--|
| DockingSamples | Shows a docking window infrastructure. |

DockControl 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 C1DockControl in general. If you are unfamiliar with the **ComponentOne DockControl for WPF** product, please see the [DockControl for WPF Quick Start](#) (page 17) first.

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

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

Setting the Dock Mode

You can set the dock mode at design time using the DockMode property.

To set the dock mode, follow these steps:

1. Open the .xaml page in Visual Studio.
2. Place your cursor between the `<Grid></Grid>` tags.
3. In the Toolbox, double-click the C1DockControl icon to add the control to the project.
4. Place your cursor between the `<c1:C1DockControl>` and `</c1:C1DockControl>` tags.

- In the Toolbox, double-click the C1DockTabControl icon to add the control to the project.
- Set the DockMode property to **Sliding**. Your XAML markup will now look similar to this:

```
<c1:C1DockControl>  
    <c1:C1DockTabControl DockMode="Sliding"></c1:C1DockTabControl>  
</c1:C1DockControl>
```

- Place your cursor between the `<c1:C1DockTabControl>` and `</c1:C1DockTabControl>` tags.
- In the Toolbox, double-click the C1DockTabItem icon to add the control to the project and set the **C1DockTabItem.Header** property to **Toolbox**. Your XAML should look similar to the following:

```
<c1:C1DockControl>  
    <c1:C1DockTabControl DockMode="Sliding">  
        <c1:C1DockTabItem Header="Toolbox"></c1:C1DockTabItem>  
    </c1:C1DockTabControl>  
</c1:C1DockControl>
```

- Run your project. The C1DockControl will resemble the following image:



- Click the **Toolbox** tab to slide the window into view.

