

---

ComponentOne

# PDF for Silverlight

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 PDF for Silverlight .....	1
Installing Studio for Silverlight .....	1
Studio for Silverlight Setup Files .....	1
Using Maps Powered by Esri .....	1
System Requirements .....	2
Installing Demonstration Versions .....	3
Uninstalling Studio for Silverlight .....	3
End-User License Agreement .....	3
Licensing FAQs .....	3
What is Licensing? .....	3
Studio for Silverlight Licensing .....	3
Technical Support .....	5
Redistributable Files .....	6
About This Documentation .....	6
C1.Silverlight.Pdf.dll .....	7
Studio for Silverlight Samples .....	7
C1.Silverlight.Pdf .....	7
ControlExplorer Sample .....	7
Factories Sample .....	8
Olympics Sample .....	8
QuickStart Sample .....	8
SamplesCommon Sample .....	8
SilverTunes Sample .....	8
StockPortfolio Sample .....	8
Templates Sample .....	9
Introduction to Silverlight .....	11
Silverlight Resources .....	11
Creating a New Silverlight Project .....	12
Theming .....	15
Available Themes .....	15

BureauBlack .....	15
ExpressionDark .....	16
ExpressionLight .....	17
RainierOrange .....	17
ShinyBlue .....	18
WhistlerBlue .....	19
Custom Themes .....	19
Included XAML Files .....	19
C1.Silverlight .....	19
C1.Silverlight.Chart .....	21
C1.Silverlight.Chart.Editor .....	22
C1.Silverlight.Chart .....	22
C1.Silverlight.DataGrid .....	22
C1.Silverlight.DataGrid.Ria .....	23
C1.Silverlight.DateTimeEditors .....	23
C1.Silverlight.Docking .....	23
C1.Silverlight.Extended .....	23
C1.Silverlight.Gauge .....	24
C1.Silverlight.Imaging .....	24
C1.Silverlight.Legacy .....	24
C1.Silverlight.Maps .....	24
C1.Silverlight.MediaPlayer .....	25
C1.Silverlight.PdfViewer .....	25
C1.Silverlight.ReportViewer .....	25
C1.Silverlight.RichTextBox .....	25
C1.Silverlight.RichTextBox.Toolbar .....	25
C1.Silverlight.Schedule .....	25
C1.Silverlight.SpellChecker .....	26
C1.Silverlight.Theming.BureauBlack .....	26
C1.Silverlight.Theming.ExpressionDark .....	27
C1.Silverlight.Theming.ExpressionLight .....	27
C1.Silverlight.Theming.RainierOrange .....	27
C1.Silverlight.Theming.ShinyBlue .....	27
C1.Silverlight.Theming.WhistlerBlue .....	27
C1.Silverlight.Toolbar .....	28
Implicit and Explicit Styles .....	28
Implicit Styles .....	28

WPF and Silverlight Styling .....	28
Using the ImplicitStyleManager.....	29
Applying Themes to Controls.....	30
Applying Themes to an Application .....	31
ComponentOne ClearStyle Technology .....	34
How ClearStyle Works .....	34
ClearStyle Properties .....	34
PDF for Silverlight .....	37
PDF for Silverlight Features .....	37
PDF for Silverlight Quick Start .....	38
Step 1 of 4: Creating an Application with the C1PdfDocument Object .....	38
Step 2 of 4: Adding Content to the Page.....	39
Step 3 of 4: Saving the document .....	39
Step 4 of 4: Running the Application .....	40
Using ComponentOne PDF for Silverlight .....	41
Adding Text .....	41
Drawing Text.....	41
Measuring Text.....	42
Making Text Flow from Page to Page .....	43
Adding Images .....	44
Adding Graphics.....	46
Creating Pages and Overlays .....	48
Adding Bookmarks to a PDF Document.....	51
Adding Links to a PDF Document .....	52
Attaching Files to a PDF Document.....	53
Applying Security and Permissions .....	55
Using Metafiles .....	56



# ComponentOne PDF for Silverlight

Since printing support is not included in Silverlight, we give you the ultimate solution: **ComponentOne PDF™ for Silverlight**. Easily create, print and email Adobe PDF documents from your apps. **PDF for Silverlight** gives you security, compression, outlining, hyper-linking, and attachments.

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

## Installing Studio for Silverlight

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

### Studio for Silverlight Setup Files

The **ComponentOne Studio for Silverlight** installation program will create the following directory: **C:\Program Files\ComponentOne\Studio for Silverlight 4.0**. This directory contains the following subdirectories:

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

### Samples

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

**Windows XP path:** C:\Documents and Settings\\My Documents\ComponentOne Samples\Studio for Silverlight 4.0

**Windows Vista and Windows 7 path:** C:\Users\\Documents\ComponentOne Samples\Studio for Silverlight 4.0

See the [Studio for Silverlight Samples](#) (page 7) topic for more information about each sample.

### 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 1) or visit the Esri website at <http://www.esri.com> for additional information.

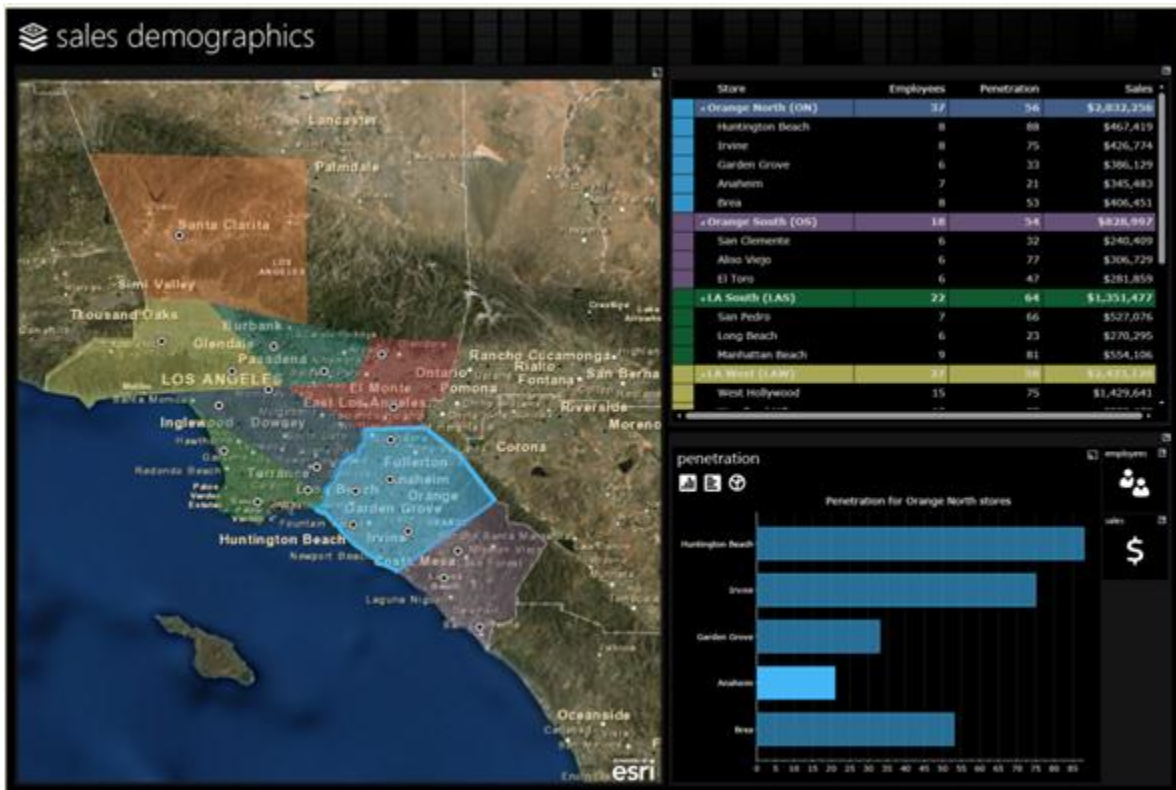
### Using Maps Powered by Esri

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

## System Requirements

System requirements for **ComponentOne Studio for Silverlight** include the following:

- Microsoft Silverlight 4.0 or later
- Microsoft Visual Studio 2008 or later

## Installing Demonstration Versions

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

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

## Uninstalling Studio for Silverlight

To uninstall **ComponentOne Studio for Silverlight**:

1. Open the **Control Panel** and select **Add or Remove Programs** (XP) or **Programs and Features** (Windows 7/Vista).
2. Select **ComponentOne Studio for Silverlight 4.0** 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

The **ComponentOne Studio for Silverlight** product is a commercial product. It is not shareware, freeware, or open source. If you use it in production applications, please purchase a copy from our Web site or from the software reseller of your choice.

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 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.

### Studio for Silverlight Licensing

Licensing for **ComponentOne Studio for Silverlight** is similar to licensing in other ComponentOne products but there are a few differences to note.

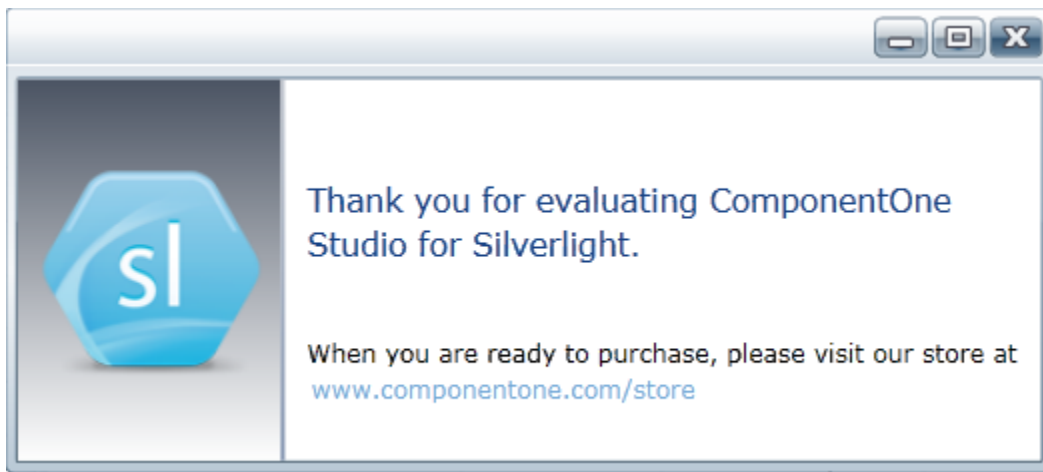
Initially licensing is handled similarly to other ComponentOne products. When a user decides to purchase a product, he receives an installation program and a Serial Number. During the installation process, the user is prompted for the serial number that is saved on the system.

In **ComponentOne Studio for Silverlight**, when a control is dropped on a form, a license nag dialog box appears one time. The nag screen appears similar to the following image:

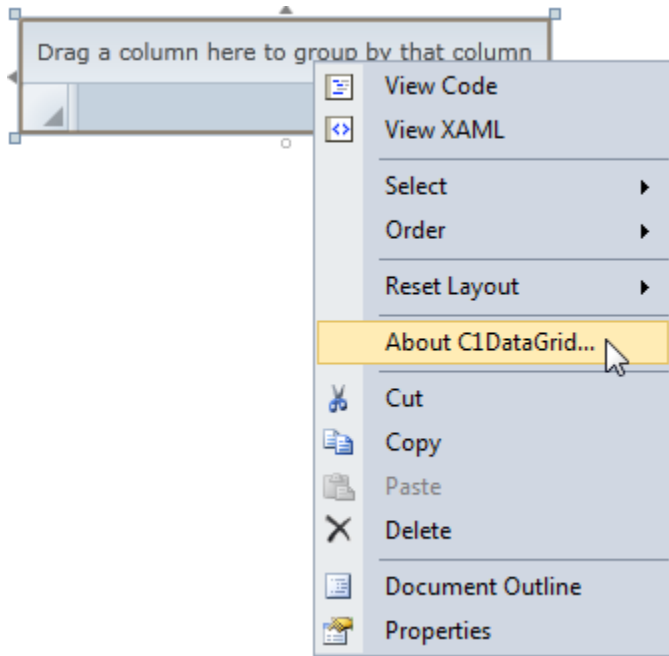


The **About** dialog box displays version information, online resources, and (if the control is unlicensed) buttons to purchase, activate, and register the product.

All ComponentOne products are designed to display licensing information at run time if the product is not licensed. None will throw licensing exceptions and prevent applications from running. Each time an unlicensed Silverlight application is run; end-users will see the following pop-up dialog box:



To stop this message from appearing, enter the product's serial number by clicking the **Activate** button on the **About** dialog box of any ComponentOne product, if available, or by rerunning the installation and entering the serial number in the licensing dialog box. To open the **About** dialog box, right-click the control and select the **About** option:



Note that when the user modifies any property of a ComponentOne Silverlight control in Visual Studio or Blend, the product will check if a valid license is present. If the product is not currently licensed, an attached property will be added to the control (the **C1NagScreen.Nag** property). Then, when the application executed, the product will check if that property is set, and show a nag screen if the **C1NagScreen.Nag** property is set to **True**. If the user has a valid license the property is not added or is just removed.

One important aspect of this of this process is that the user should manually remove all instances of **c1:C1NagScreen.Nag="true"** in the XAML markup in all files after registering the license (or re-open all the files that include ComponentOne controls in any of the editors). This will ensure that the nag screen does not appear when the application is run.

## 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 Studio for Silverlight** 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.Silverlight.dll
- C1.Silverlight.Pdf.dll

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

## About This Documentation

### Acknowledgements

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

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](#).

## C1.Silverlight.Pdf.dll

The **C1.Silverlight.Pdf.dll** assembly contains the **C1Pdf** component. Since printing support is not included in Silverlight, we give you the ultimate solution: **ComponentOne PDF™ for Silverlight**. Easily create, print and email Adobe PDF documents from your apps. PDF for Silverlight gives you security, compression, outlining, hyper-linking, and attachments.

### Main Classes

The following main classes are included in the **C1.Silverlight.Pdf.dll** assembly:

- **C1Pdf: ComponentOne PDF™ for Silverlight** allows you to easily create, print and email Adobe PDF documents from your apps. PDF for Silverlight gives you security, compression, outlining, hyper-linking, and attachments.

## Studio for Silverlight Samples

If you just installed **ComponentOne Studio for Silverlight**, open Visual Studio 2008 and load the **Samples.sln** solution located in the **C:\Documents and Settings\<username>\My Documents\ComponentOne Samples\Studio for Silverlight** or **C:\Users\<username>\Documents\ComponentOne Samples\Studio for Silverlight** folder. This solution contains all the samples that ship with this release. Each sample has a readme.txt file that describes it and two projects named as follows:

<SampleName>	Silverlight project (client-side project)
<SampleName>Web	ASP.NET project that hosts the Silverlight project (server-side project)

To run a sample, right-click the **<SampleName>Web** project in the Solution Explorer, select **Set as Startup Project**, and press F5.

The following topics, organized by folder, describe each of the included samples.

### C1.Silverlight.Pdf

The following sample is installed in the **C1.Silverlight.Pdf** folder in the samples directory by default.

#### ***PdfCreator Sample***

The **PdfCreator** sample is installed in the **C1.Silverlight.Pdf\PdfCreator** folder in the samples directory.

This sample demonstrates how to use the **C1Pdf** control to add PDF support to a Silverlight application.

#### **ControlExplorer Sample**

The **ControlExplorer** sample is installed in the **ControlExplorer** folder in the samples directory.

This sample displays a tree with all C1.Silverlight controls and demonstrates them in action. The application has two panes. The left pane contains a tree that lists all controls in the C1.Silverlight library. Select a control on the tree and an instance appears on the right pane, along with a few properties that you can modify to see how they affect the control.

The projects show all controls in the C1.Silverlight library, and shows how to use reflection to create controls and interact with them dynamically. The project is driven by an XML resource file that lists the controls that should be exposed by the sample. The DataGrid sample uses a data source that is a DataSet stored as a zip-compressed embedded resource.

## Factories Sample

The **Factories** sample is installed in the **Factories** folder in the samples directory.

This sample demonstrates how to use **CIMaps** to show production and distribution information from a manufacturer. There are three types of location marked in the map: factories, offices and stores. Each has a small icon providing related information, click on it to view an expanded version. Stores are only visible by zooming in near an office. The user can position new locations by dragging them from the top-right toolbar to the map.

## Olympics Sample

The **Olympics** sample is installed in the **Olympics** folder in the samples directory.

This sample allows you to check out the performance of any country in the Olympic Games. The sample was built during the 2008 Olympic Games in Beijing, and got a lot of traffic during the games. The sample used a Web service to retrieve the number of medals won by each country in real time, and then displayed that information in three different ways: map, chart, and grid.

## QuickStart Sample

The **Factories** sample is installed in the **QuickStart** folder in the samples directory.

This sample demonstrates how to instantiate and use the several of the controls in **ComponentOne Studio for Silverlight**.

## SamplesCommon Sample

The **SamplesCommon** project is installed in the **SamplesCommon\SamplesCommon** folder in the samples directory. This sample project includes elements that are incorporated in other sample projects.

## SilverTunes Sample

The **SilverTunes** sample is installed in the **SilverTunes** folder in the samples directory.

This sample showcases a Silverlight implementation of an 'iTunes' like application. The sample loads artist, album, and song information from a database. It then shows the album covers in a carousel, with a rotating effect and reflection. The user can search by artist, album, and song. The sample includes a few MP3files so the user can play some of the songs. The MP3 files included in the sample are provided for demonstration purposes only and are truncated to protect the copyright owners.

## StockPortfolio Sample

The **StockPortfolio** sample is installed in the **StockPortfolio** folder in the samples directory.

This sample demonstrates a stock portfolio application. This sample was created for the ReMIX 07 in Boston, to demonstrate a business application in Silverlight. When it starts, the sample loads the stock portfolio from isolated storage (or creates a new one when run for the first time). Once the portfolio is loaded, the application retrieves real-time stock quotes using a web service and shows the portfolio information in a grid. The grid shows the personal portfolio information (stocks, quantities, price paid), real time stock information (last traded price, day high and low), and some calculated data (current market value, gain/loss).

The user can add new stocks to the portfolio or remove existing ones. When the application exits, the current portfolio information is saved to isolated storage so it is available next time the application runs. The user can also see the historical data for companies in the portfolio. Double-clicking a symbol on the grid shows a composite

chart with detailed information on the top and a zoom chart below. The user can slide and expand the zoom window in the bottom chart to see details on the top chart.

## Templates Sample

The **Templates** sample is installed in the **Templates** folder in the samples directory.

This sample shows how to use Data Templates to determine how data is displayed in controls. The sample creates a list of items and applies the same data template to two controls of different types (a **ListBox** and a **ComboBox**). See the **Data Templates** topic for more information.



# Introduction to Silverlight

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

## Silverlight Resources

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

- <http://www.silverlight.net>  
The official Silverlight site, with many links to downloads, samples, tutorials, and more.
- <http://silverlight.net/learn/tutorials.aspx>  
Silverlight tutorials by Jesse Liberty. Topics covered include:
  - Tutorial 1: Silverlight User Interface Controls
  - Tutorial 2: Data Binding
  - Tutorial 3: Displaying SQL Database Data in a DataGrid using LINQ and WCF
  - Tutorial 4: User Controls
  - Tutorial 5: Styles, Templates and Visual State Manager
  - Tutorial 6: Expression Blend for Developers
  - Tutorial 7: DataBinding & DataTemplates Using Expression Blend
  - Tutorial 8: Multi-page Applications
  - Tutorial 9: ADO.NET DataEntities and WCF Feeding a Silverlight DataGrid
  - Tutorial 10: Hyper-Video
- <http://timheuer.com/blog/articles/getting-started-with-silverlight-development.aspx>  
Silverlight tutorials by Tim Heuer. Topics covered include:
  - Part 1: Really getting started – the tools you need and getting your first Hello World
  - Part 2: Defining UI Layout – understanding layout and using Blend to help
  - Part 3: Accessing data – how to get data from where
  - Part 4: Binding the data – once you get the data, how can you use it?
  - Part 5: Integrating additional controls – using controls that aren't a part of the core
  - Part 6: Polishing the UI with styles and templates
  - Part 7: Taking the application out-of-browser
- <http://weblogs.asp.net/scottgu/pages/silverlight-posts.aspx>  
Scott Guthrie's Silverlight Tips, Tricks, Tutorials and Links Page. A useful resource, this page links to several tutorials and samples.
- <http://weblogs.asp.net/scottgu/archive/2008/02/22/first-look-at-silverlight-2.aspx>  
An excellent eight-part tutorial by Scott Guthrie, covering the following topics:

- Part 1: Creating "Hello World" with Silverlight 2 and VS 2008
- Part 2: Using Layout Management
- Part 3: Using Networking to Retrieve Data and Populate a DataGrid
- Part 4: Using Style Elements to Better Encapsulate Look and Feel
- Part 5: Using the ListBox and DataBinding to Display List Data
- Part 6: Using User Controls to Implement Master/Details Scenarios
- Part 7: Using Templates to Customize Control Look and Feel
- Part 8: Creating a Digg Desktop Version of our Application using WPF
- <http://blogs.msdn.com/corrinab/archive/2008/03/11/silverlight-2-control-skins.aspx>  
A practical discussion of skinning Silverlight controls and applications by Corrina Barber.

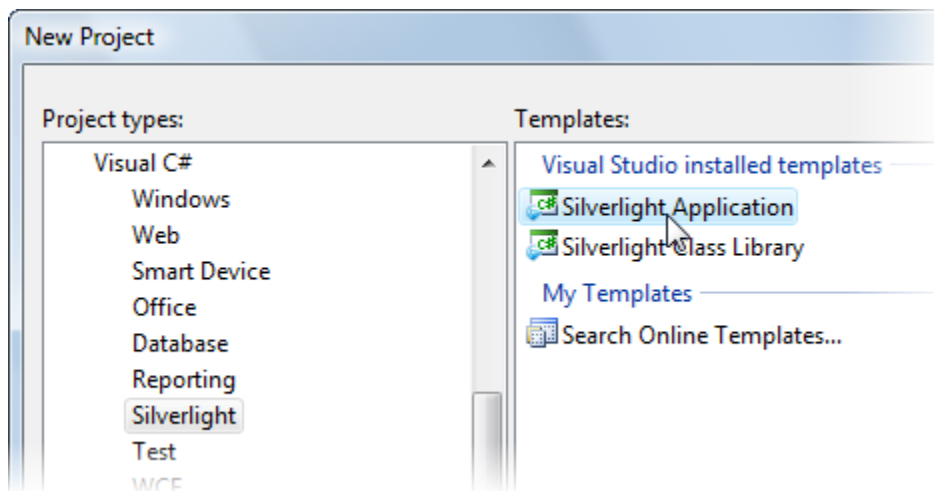
## Creating a New Silverlight Project

The following topic details how to create a new Silverlight project in Microsoft Visual Studio 2008 and in Microsoft Expression Blend 3.

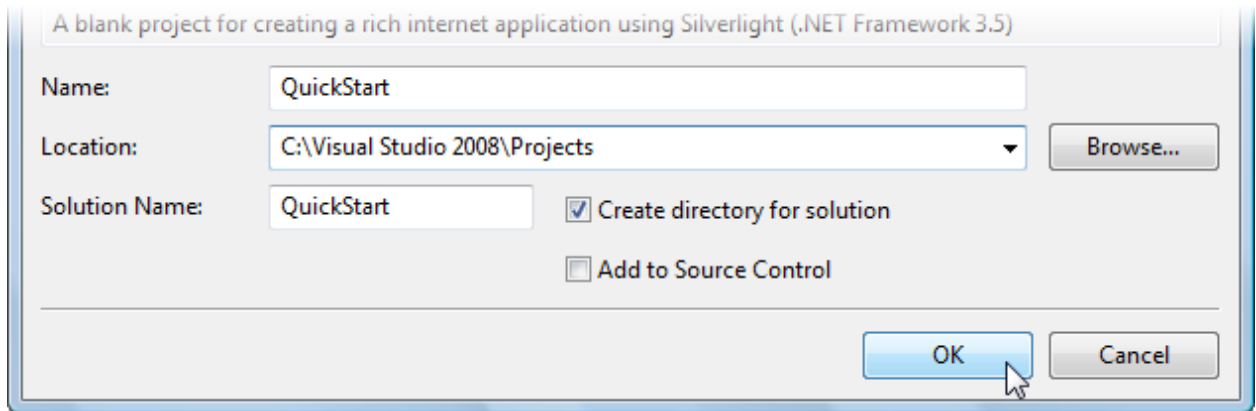
### In Visual Studio 2008

Complete the following steps to create a new Silverlight project in Microsoft Visual Studio 2008:

1. Select **File | New | Project** to open the **New Project** dialog box in Visual Studio 2008.
2. In the **Project types** pane, expand either the **Visual Basic** or **Visual C#** node and select **Silverlight**.
3. Choose **Silverlight Application** in the **Templates** pane.

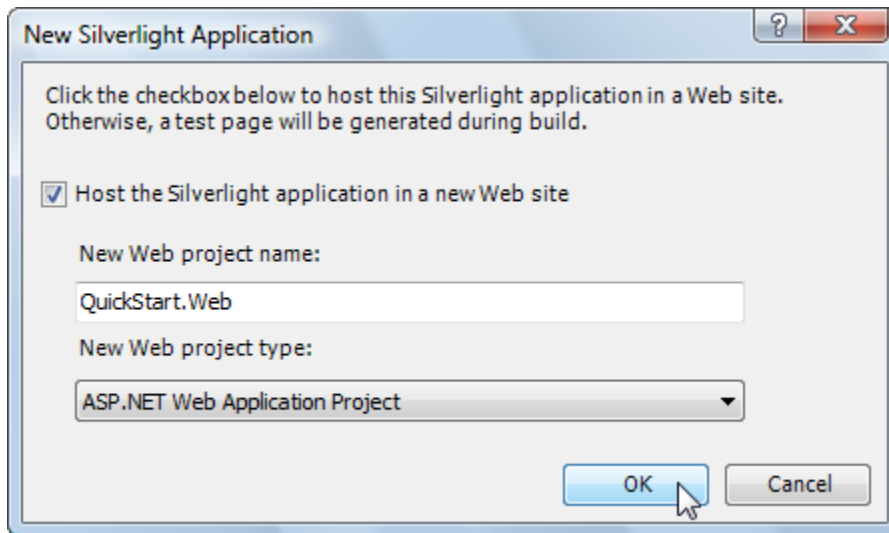


4. Name the project, specify a location for the project, and click **OK**.



Next, Visual Studio will prompt you for the type of hosting you want to use for the new project.

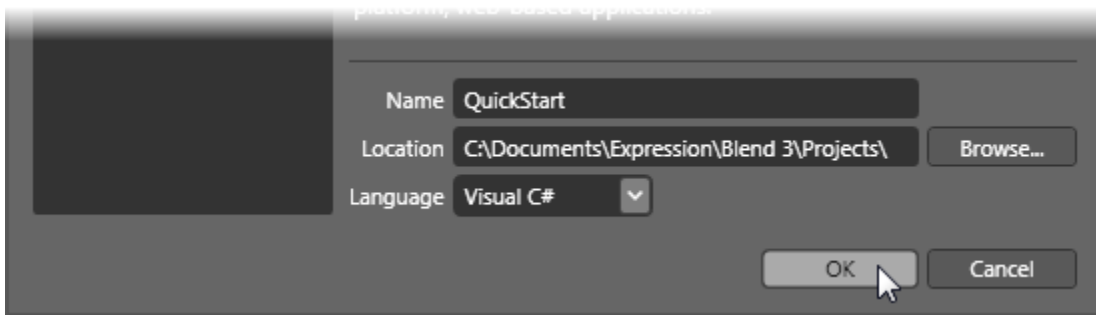
5. In the **New Silverlight Application** dialog box, select **OK** to accept the default name and options and to create the project.



#### In Expression Blend 4

Complete the following steps to create a new Silverlight project in Microsoft Expression Blend 4:

1. Select **File | New Project** to open the **New Project** dialog box in Blend 4.
2. In the **Project types** pane, click the **Silverlight** node.
3. In the right pane, choose **Silverlight Application + Website** in the **Templates** pane to create a project with an associated Web site.
4. Name the project, specify a location for the project, choose a language (**Visual C#** or **Visual Basic**), and click **OK**.



Your new project will be created.

### The Project

The solution you just created will contain two projects, **YourProject** and **YourProject.Web**:

- **YourProject**: This is the Silverlight application proper. It will produce a XAP file that gets downloaded to the client and runs inside the Silverlight plug-in.
- **YourProject.Web**: This is the host application. It runs on the server and provides support for the Silverlight application.

# Theming

One of the main advantages to using Silverlight is the ability to change the style or template of any control. Controls are "lookless" with fully customizable user interfaces and the ability to use built-in and custom themes. Themes allow you to customize the appearance of controls and take advantage of Silverlight's XAML-based styling. The following topics introduce you to styling Silverlight controls with themes.

You can customize WPF and Microsoft Silverlight controls by creating and modifying control templates and styles. This results in a unique and consistent look for your application.

Templates and styles define the pieces that make up a control and the default behavior of the control, respectively. You can create templates and styles by making copies of the original styles and templates for a control. Modifying templates and styles is an easy way to essentially make new controls in Design view of Microsoft Expression Blend, without having to use code. The following topics provide a detailed comparison of styles and templates to help you decide whether you want to modify the style or template of a control, or both. The topics also discuss the built-in themes available in **ComponentOne Studio for Silverlight**.

## Available Themes

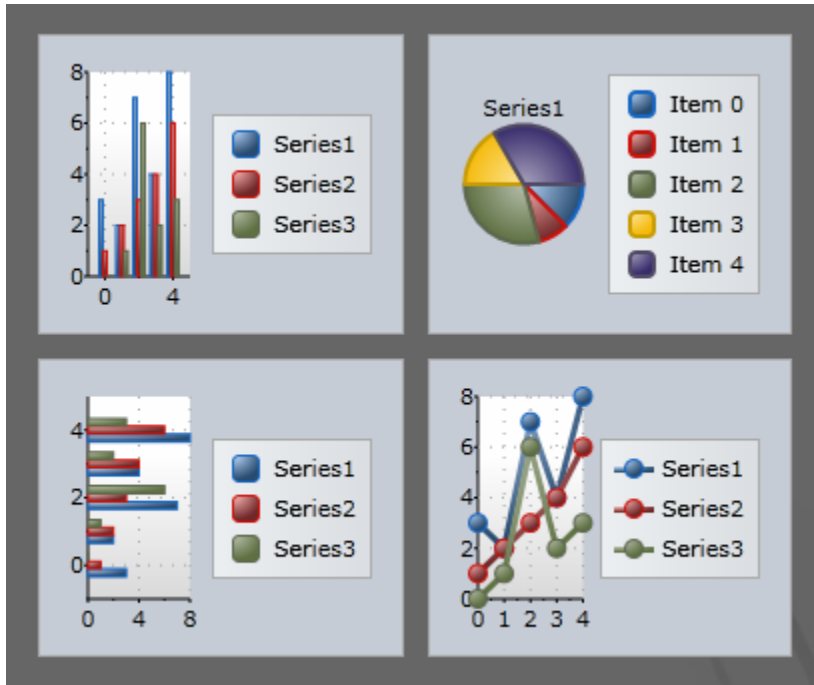
**ComponentOne Studio for Silverlight** includes several theming options, and several built-in Silverlight Toolkit themes including:

- BureauBlack
- ExpressionDark
- ExpressionLight
- RainierOrange
- ShinyBlue
- WhistlerBlue

Each of these themes is based on themes in the Silverlight Toolkit and installed in its own assembly in the **Studio for Silverlight** installation directory. The following topics detail each built-in theme.

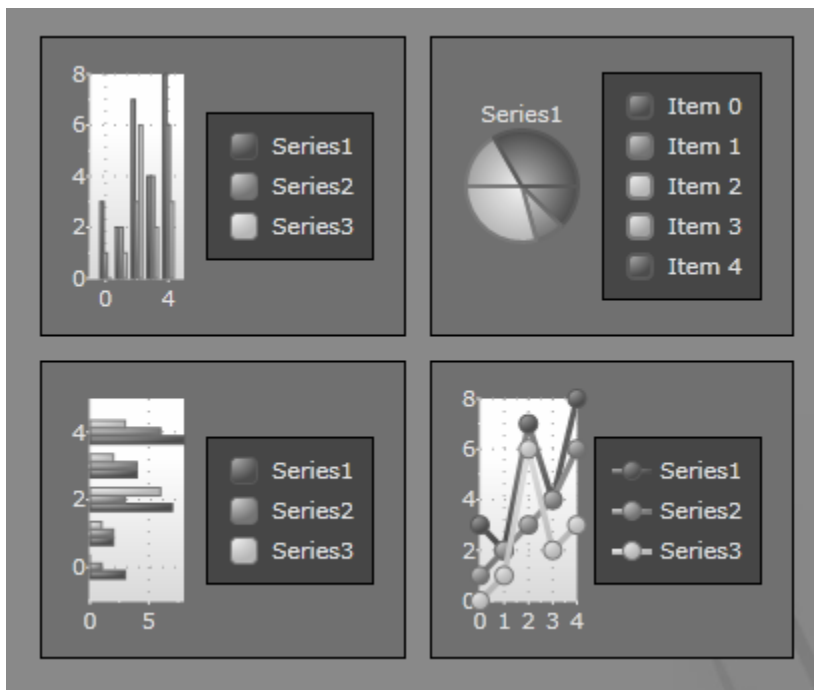
### BureauBlack

The BureauBlack theme is a dark colored theme similar to the Microsoft Bureau Black theme included in the Silverlight Toolkit. The BureauBlack theme appears similar the following when applied to the **ComponentOne Studio for Silverlight** charting controls:



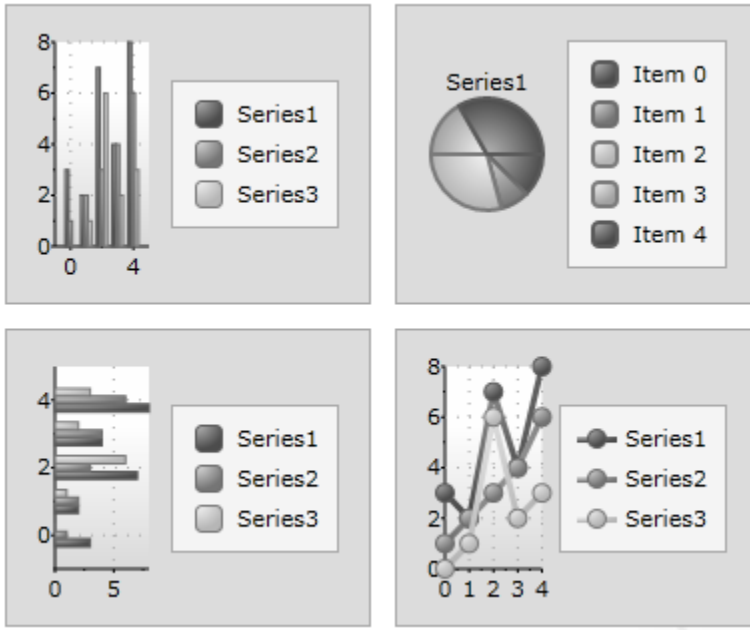
### ExpressionDark

The ExpressionDark theme is a grayscale theme based on the Microsoft Expression Dark theme, which is included in the Silverlight Toolkit. For example, the theme appears similar the following when applied to the **ComponentOne Studio for Silverlight** charting controls:



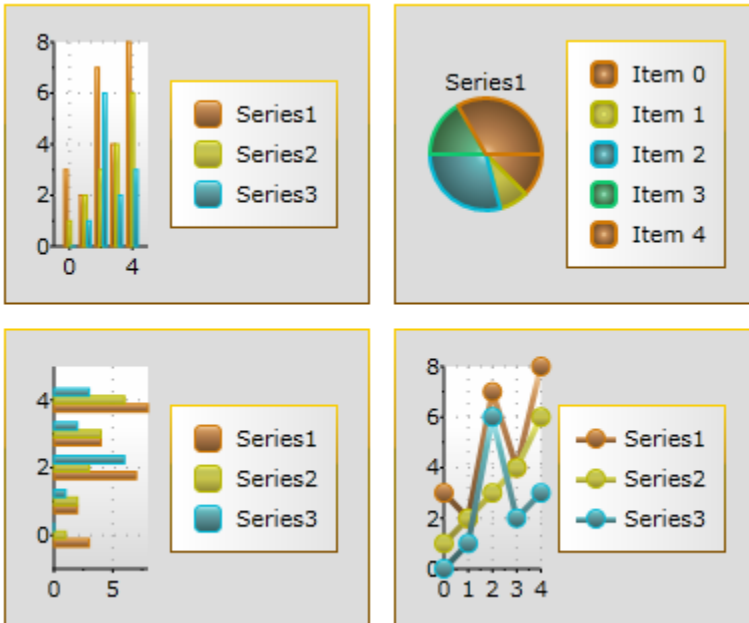
## ExpressionLight

The ExpressionLight theme is a grayscale theme based on the Microsoft Expression Light theme, which is included in the Silverlight Toolkit. For example, the theme appears similar the following when applied to the **ComponentOne Studio for Silverlight** charting controls:



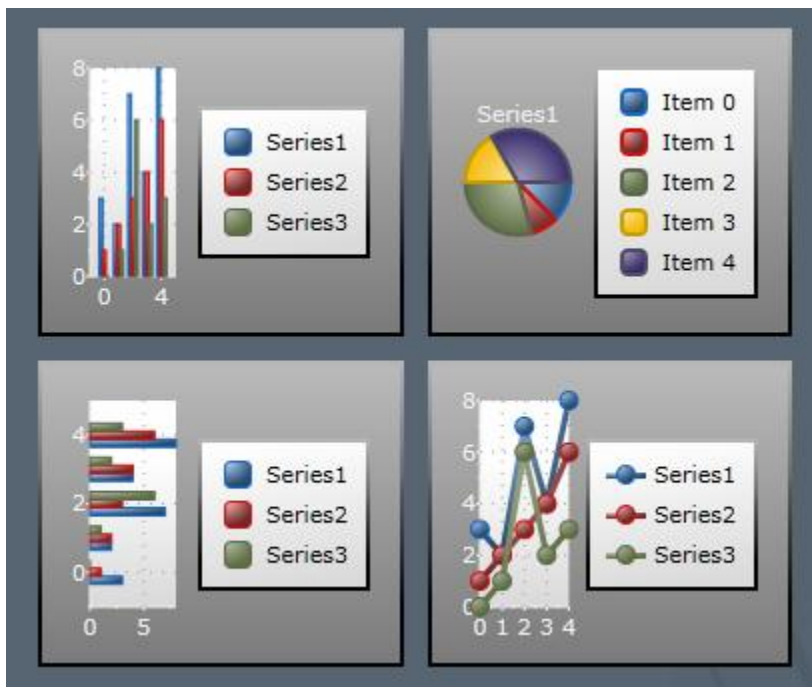
## RainierOrange

The RainierOrange theme is an orange-based theme similar to the Microsoft Rainier Orange theme, which is included in the Silverlight Toolkit. The RainierOrange theme appears similar the following when applied to the **ComponentOne Studio for Silverlight** charting controls:



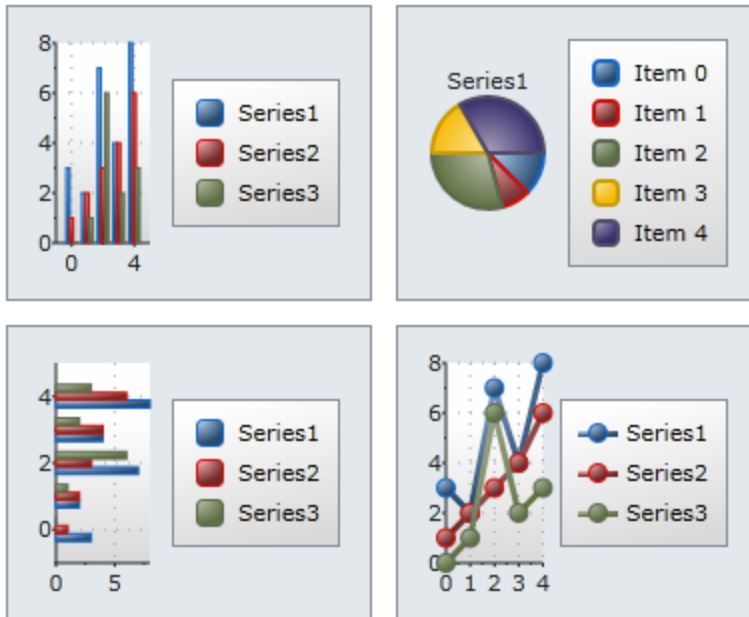
## ShinyBlue

The ShinyBlue theme is a blue-based theme similar to the Microsoft Shiny Blue theme included in the Silverlight Toolkit. The ShinyBlue theme appears similar the following when applied to the **ComponentOne Studio for Silverlight** charting controls:



## WhistlerBlue

The WhistlerBlue theme is a blue-based theme similar to the Microsoft Whistler Blue theme, which is included in the Silverlight Toolkit. The WhistlerBlue theme appears similar the following when applied to the **ComponentOne Studio for Silverlight** charting controls:



## Custom Themes

In addition to using one of the built-in themes, you can create your own custom theme from scratch or create a custom theme based on an existing built-in theme. See [Included XAML Files](#) (page 19) for the included files that you can base a theme on.

## Included XAML Files

Several auxiliary XAML elements are installed with **ComponentOne Studio for Silverlight**. These elements include templates and themes and are located in the **Studio for Silverlight** installation directory. You can incorporate these elements into your project to, for example, create your own theme based on the included Office 2007 themes.

By default, these files are located in the **generics.zip** file in the **C:\Program Files\ComponentOne\Studio for Silverlight 4.0\Help** folder. Unzip the **generics.zip** file to a folder to see all the XAML files associated with **Studio for Silverlight** controls. In the following topics the included files are listed by assembly with their location folder within the **generics.zip** file noted.

### C1.Silverlight

The following XAML files can be used to customize items in the **C1.Silverlight** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight\themes	Specifies the templates for different styles and

		the initial style of the controls.
Common.xaml	C1.Silverlight\themes	Specifies attributes for common elements in the controls.
C1Button.xaml	C1.Silverlight\themes	Specifies attributes for C1Button.
C1ComboBox.xaml	C1.Silverlight\themes	Specifies attributes for C1ComboBox.
C1DropDown.xaml	C1.Silverlight\themes	Specifies attributes for C1DropDown.
C1FilePicker.xaml	C1.Silverlight\themes	Specifies attributes for C1FilePicker.
C1HeaderedContentControl.xaml	C1.Silverlight\themes	Specifies attributes for C1HeaderedContentControl.
C1LayoutTransformer.xml	C1.Silverlight\themes	Specifies attributes for C1LayoutTransformer.
C1Menu.xaml	C1.Silverlight\themes	Specifies attributes for C1Menu.
C1NumericBox.xaml	C1.Silverlight\themes	Specifies attributes for C1NumericBox.
C1RangeSlider.xaml	C1.Silverlight\themes	Specifies attributes for C1RangeSlider.
C1ScrollBar.xaml	C1.Silverlight\themes	Specifies attributes for C1ScrollBar.
C1ScrollViewer.xaml	C1.Silverlight\themes	Specifies attributes for C1ScrollViewer.
C1Separator.xaml	C1.Silverlight\themes	Specifies attributes for C1Separator.
C1TabControl.xaml	C1.Silverlight\themes	Specifies attributes for C1TabControl.
C1TextBoxBase.xaml	C1.Silverlight\themes	Specifies attributes for C1TextBoxBase.
C1TextEditableContentControl.xaml	C1.Silverlight\themes	Specifies attributes for C1TextEditableContentControl.
C1TreeView.xaml	C1.Silverlight\themes	Specifies attributes for C1TreeView.
C1ValidationDecorator.xml	C1.Silverlight\themes	Specifies attributes for C1ValidationDecorator.
generic.xaml	\C1.Silverlight\Phone\themes	Specifies the templates for the Metro theme for the controls.
Common.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for common elements in the controls.
C1Button.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1Button.
C1ComboBox.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1ComboBox.
C1DropDown.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1DropDown.
C1FilePicker.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1FilePicker.
C1HeaderedContentControl.Metro.xml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1HeaderedContentControl.
C1LayoutTransformer.Metro.xml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1LayoutTransformer.
C1Menu.Metro.xml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1Menu.
C1NumericBox.Metro.xml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1NumericBox.
C1RangeSlider.Metro.xml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1RangeSlider.

C1ScrollBar.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1ScrollBar.
C1ScrollViewer.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1ScrollViewer.
C1TabControl.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1TabControl.
C1TextBoxBase.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1TextBoxBase.
C1TextEditableContentControl.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1TextEditableContentControl.
C1TreeView.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1TreeView.
C1ValidationDecorator.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1ValidationDecorator.
C1Window.Metro.xaml	\C1.Silverlight\Phone\themes	Specifies attributes for the Metro theme for C1Window.

## C1.Silverlight.Chart

The following XAML files can be used to customize items in the **C1.Silverlight.Chart** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Chart\themes	Specifies the templates for different styles and the initial style of the chart.
DuskBlue.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the DuskBlue theme.
DuskGreen.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the DuskGreen theme.
MediaPlayer.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the MediaPlayer theme.
Office2003Blue.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2003Blue theme.
Office2003Classic.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2003Classic theme.
Office2003Olive.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2003Olive theme.
Office2003Royale.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2003Royale theme.
Office2003Silver.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2003Silver theme.
Office2007Black.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2007Black theme.
Office2007Blue.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2007Blue theme.
Office2007Silver.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Office2007Silver theme.
Vista.xaml	C1.Silverlight.Chart\ThemesSL	Specifies the attributes for the Vista theme.
generic.xaml	C1.Silverlight.Chart\Phone	Specifies the templates for the Metro theme for

	hone\themes	the controls.
--	-------------	---------------

## C1.Silverlight.Chart.Editor

The following XAML files can be used to customize items in the **C1.Silverlight.Chart.Editor** assembly:

Element	Folder	Description
AxisEditor.xaml	C1.Silverlight.Chart.Editor	Specifies the attributes for the Axis Editor.
ChartEditor.xaml	C1.Silverlight.Chart.Editor	Specifies the attributes for the Chart Editor.
DataLabelEditor.xaml	C1.Silverlight.Chart.Editor	Specifies the attributes for the Data Label Editor.
LegendEditor.xaml	C1.Silverlight.Chart.Editor	Specifies the attributes for the Legend Editor.
DashesEditor.xaml	C1.Silverlight.Chart.Editor\AuxControls	Specifies the attributes for the Dashes Editor.
PropertyEditor.xaml	C1.Silverlight.Chart.Editor\PropertyEditors	Specifies the attributes for the Property Editor.

## C1.Silverlight.Chart

The following XAML files can be used to customize items in the **C1.Silverlight.Chart3D** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Chart3D\themes	Specifies the templates for different styles and the initial style of the chart.

## C1.Silverlight.DataGrid

The following XAML file can be used to customize items in the **C1.Silverlight.DataGrid** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.DataGrid\themes	Specifies the templates for different styles and the initial style of the controls.
Common.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for common elements in the controls.
DataGridCellPresenter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for common elements in the controls.
DataGridColumnHeaderPresenter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for the column header presenter.
DataGridDetailsPresenter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for the data details presenter.
DataGridDragNDrop.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for grid drag-and-drop operation.
DataGridFilter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for the grid's filtering.
DataGridGroupingPresenter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for the grouping presenter.

DataGridRowHeaderPresenter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for the row header presenter.
DataGridRowPresenter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for the row presenter.
DataGridVerticalFreezingSeparatorPresenter.xaml	C1.Silverlight.DataGrid\themes	Specifies attributes for the freezing separator presenter.

## C1.Silverlight.DataGrid.Ria

The following XAML file can be used to customize items in the **C1.Silverlight.DataGrid.Ria** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.DataGrid.Ria\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.DateTimeEditors

The following XAML file can be used to customize items in the **C1.Silverlight.DateTimeEditors** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.DateTimeEditors\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.Docking

The following XAML file can be used to customize items in the **C1.Silverlight.Docking** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Docking\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.Extended

The following XAML file can be used to customize items in the **C1.Silverlight.Extended** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Extended\themes	Specifies the templates for different styles and the initial style of the controls.
C1Accordion.xaml	C1.Silverlight.Extended\themes	Specifies attributes for C1Accordion.
C1Book.xaml	C1.Silverlight.Extended\themes	Specifies attributes for C1Book.
C1ColorPicker.xaml	C1.Silverlight.Extended\themes	Specifies attributes for C1ColorPicker.
C1CoverFlow.xaml	C1.Silverlight.Extended\themes	Specifies attributes for C1CoverFlow.
C1Expander.xaml	C1.Silverlight.Extended\themes	Specifies attributes for C1Expander.
C1PropertyGrid.xaml	C1.Silverlight.Extended\themes	Specifies attributes for C1PropertyGrid.

C1Reflector.xaml	C1.Silverlight.Extended\themes	Specifies attributes for C1Reflector.
generic.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies the templates for the Metro theme of the controls.
C1Accordion.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies attributes for the Metro theme for C1Accordion.
C1Book.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies attributes for the Metro theme for C1Book.
C1ColorPicker.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies attributes for the Metro theme for C1ColorPicker.
C1CoverFlow.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies attributes for the Metro theme for C1CoverFlow.
C1Expander.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies attributes for the Metro theme for C1Expander.
C1PropertyGrid.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies attributes for the Metro theme for C1PropertyGrid.
C1Reflector.xaml	C1.Silverlight.Extended\Phone\Themes	Specifies attributes for the Metro theme for C1Reflector.

## C1.Silverlight.Gauge

The following XAML file can be used to customize items in the **C1.Silverlight.Gauge** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Gauge\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.Imaging

The following XAML file can be used to customize items in the **C1.Silverlight.Imaging** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Imaging\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.Legacy

The following XAML file can be used to customize items in the **C1.Silverlight.Legacy** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Legacy\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.Maps

The following XAML file can be used to customize items in the **C1.Silverlight.Maps** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Maps\themes	Specifies the templates for different styles and the initial style of the controls.

ZoomScrollBar.xaml	C1.Silverlight.Maps\themes	Specifies attributes for the zoom scroll bar.
--------------------	----------------------------	---

## C1.Silverlight.MediaPlayer

The following XAML file can be used to customize items in the **C1.Silverlight.MediaPlayer** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.MediaPlayer\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.PdfViewer

The following XAML file can be used to customize items in the **C1.Silverlight.PdfViewer** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.PdfViewer\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.ReportViewer

The following XAML file can be used to customize items in the **C1.Silverlight.ReportViewer** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.ReportViewer\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.RichTextBox

The following XAML file can be used to customize items in the **C1.Silverlight.RichTextBox** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.RichTextBox\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.RichTextBox.Toolbar

The following XAML file can be used to customize items in the **C1.Silverlight.RichTextBox.Toolbar** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.RichTextBox.Toolbar\themes	Specifies the templates for different styles and the initial style of the controls.

## C1.Silverlight.Schedule

The following XAML files can be used to customize items in the **C1.Silverlight.Schedule** assembly:

Element	Folder	Description
EditAppointmentControl.Sil	C1.Silverlight.Schedu	Specifies the attributes for editing appointments.

verlight.xaml	le\Dialogs	
EditCollectionControl.xaml	C1.Silverlight.Schedule\Dialogs	Specifies the attributes for editing collections.
EditRecurrenceControl.Silverlight.xaml	C1.Silverlight.Schedule\Dialogs	Specifies the attributes for editing appointment recurrence.
RecChoiceControl.Silverlight.xaml	C1.Silverlight.Schedule\Dialogs	Specifies the attributes for choosing recurrence.
SelectFromListScene.Silverlight.xaml	C1.Silverlight.Schedule\Dialogs	Specifies the attributes for resources from lists.
SelectFromListScene.WPF.xaml	C1.Silverlight.Schedule\Dialogs	Specifies the attributes for resources from lists.
ShowRemindersControl.Silverlight.xaml	C1.Silverlight.Schedule\Dialogs	Specifies the attributes for schedule reminders.
generic.xaml	C1.Silverlight.Schedule\Dialogs	Specifies the templates for different styles and the initial style of the controls.
Auxiliary.xaml	C1.Silverlight.Schedule\themes	Specifies attributes for auxiliary elements of the control.
C1Calendar.xaml	C1.Silverlight.Schedule\themes	Specifies attributes for the C1Calendar.
C1SchedulerParts.xaml	C1.Silverlight.Schedule\themes	Specifies attributes for parts of the scheduler.
Common.xaml	C1.Silverlight.Schedule\themes	Specifies attributes for common elements of the scheduler.
generic.xaml	C1.Silverlight.Schedule\themes	Specifies the templates for different styles and the initial style of the controls.
IntervalAppointmentPresenter.xaml	C1.Silverlight.Schedule\themes	Specifies attributes for the interval appointment presenter.

## C1.Silverlight.SpellChecker

The following XAML file can be used to customize items in the **C1.Silverlight.SpellChecker** assembly:

Element	Folder	Description
C1SpellDialog.xaml	C1.Silverlight.SpellChecker	Specifies the attributes for the Spell Checker Dialog Box.

## C1.Silverlight.Theming.BureauBlack

The following XAML files can be used to customize items in the **C1.Silverlight.BureauBlack** assembly:

Element	Folder	Description
BureauBlack.xaml	C1.Silverlight.Theming.g.BureauBlack	Specifies resources and styling elements for each ComponentOne Silverlight control.
System.Windows.Controls.Theming.BureauBlack.xml	C1.Silverlight.Theming.g.BureauBlack	Specifies the standard Microsoft BureauBlack resources and styling elements.
Theme.xaml	C1.Silverlight.Theming.g.BureauBlack	Specifies the standard resources and styling elements.

## C1.Silverlight.Theming.ExpressionDark

The following XAML files can be used to customize items in the **C1.Silverlight.ExpressionDark** assembly:

Element	Folder	Description
ExpressionDark.xaml	C1.Silverlight.Theming.ExpressionDark	Specifies resources and styling elements for each ComponentOne Silverlight control.
System.Windows.Controls.Theming.ExpressionDark.xaml	C1.Silverlight.Theming.ExpressionDark	Specifies the standard Microsoft ExpressionDark resources and styling elements.
Theme.xaml	C1.Silverlight.Theming.ExpressionDark	Specifies the standard resources and styling elements.

## C1.Silverlight.Theming.ExpressionLight

The following XAML files can be used to customize items in the **C1.Silverlight.ExpressionLight** assembly:

Element	Folder	Description
ExpressionLight.xaml	C1.Silverlight.Theming.ExpressionLight	Specifies resources and styling elements for each ComponentOne Silverlight control.
System.Windows.Controls.Theming.ExpressionLight.xaml	C1.Silverlight.Theming.ExpressionLight	Specifies the standard Microsoft ExpressionLight resources and styling elements.
Theme.xaml	C1.Silverlight.Theming.ExpressionLight	Specifies the standard resources and styling elements.

## C1.Silverlight.Theming.RainierOrange

The following XAML files can be used to customize items in the **C1.Silverlight.RainierOrange** assembly:

Element	Folder	Description
RainierOrange.xaml	C1.Silverlight.Theming.RainierOrange	Specifies resources and styling elements for each ComponentOne Silverlight control.
Theme.xaml	C1.Silverlight.Theming.RainierOrange	Specifies the standard resources and styling elements.

## C1.Silverlight.Theming.ShinyBlue

The following XAML files can be used to customize items in the **C1.Silverlight.ShinyBlue** assembly:

Element	Folder	Description
ShinyBlue.xaml	C1.Silverlight.Theming.ShinyBlue	Specifies resources and styling elements for each ComponentOne Silverlight control.
Theme.xaml	C1.Silverlight.Theming.ShinyBlue	Specifies the standard resources and styling elements.

## C1.Silverlight.Theming.WhistlerBlue

The following XAML files can be used to customize items in the **C1.Silverlight.WhistlerBlue** assembly:

Element	Folder	Description
WhistlerBlue.xaml	C1.Silverlight.Theming.WhistlerBlue	Specifies resources and styling elements for each ComponentOne Silverlight control.
Theme.xaml	C1.Silverlight.Theming.WhistlerBlue	Specifies the standard resources and styling elements.

## C1.Silverlight.Toolbar

The following XAML files can be used to customize items in the **C1.Silverlight.Toolbar** assembly:

Element	Folder	Description
generic.xaml	C1.Silverlight.Toolbar\themes	Specifies the templates for different styles and the initial style of the controls.
C1ToolbarTab.xaml	C1.Silverlight.Toolbar\themes	Specifies the attributes for the C1ToolbarTab.

## Implicit and Explicit Styles

The following topic detail using implicit and explicit styles and using the **ImplicitStyleManager** which is included in the Silverlight Toolkit. For more information about the Silverlight Toolkit, see [CodePlex](#).

### Implicit Styles

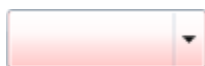
If you're familiar with WPF (Windows Presentation Foundation) you may be used to setting styles implicitly so the application has a uniform appearance – for example, you're used to setting the style for all instances of a particular control in the application's resources. Unfortunately Silverlight does not support implicit styles in the same way that WPF does and you would normally have to indicate the style to use in each instance of the control. This can be tedious to do if you have several controls on a page and that's where the **ImplicitStyleManager** comes in handy. The **ImplicitStyleManager** class is located in the Microsoft.Windows.Controls.Theming namespace (in the Microsoft.Windows.Controls assembly).

### WPF and Silverlight Styling

In WPF, you can set styles implicitly. When you set styles implicitly all instances of a particular type can be styled at once. For example, the WPF **C1DropDown** control might be styled with the following markup:

```
<Grid>
  <Grid.Resources>
    <Style TargetType="{x:Type c1:C1DropDown}">
      <Setter Property="Background" Value="Red" />
    </Style>
  </Grid.Resources>
  <c1:C1DropDown Height="30" HorizontalAlignment="Center"
    Name="C1DropDown1" VerticalAlignment="Center" Width="100" />
</Grid>
```

This would set the background of the control to be the color red as in the following image:



All **C1DropDown** controls in the grid would also appear red; **C1DropDown** controls outside of the Grid would not appear red. This is what is meant by implicit styles – the style is assigned to all controls of a particular type. Inherited controls would also inherit the style.

Silverlight, however, does not support implicit styles. In Silverlight you could add the style to the Grid's resources similarly:

```
<Grid.Resources>
    <Style x:Key="DropDownStyle" TargetType="c1:C1DropDown">
        <Setter Property="Background" Value="Red" />
    </Style>
</Grid.Resources>
```

But the Silverlight **C1DropDown** control would not be styled unless the style was explicitly set, as in the following example:

```
<c1:C1DropDown Height="30" HorizontalAlignment="Center" Name="C1DropDown1"
VerticalAlignment="Center" Width="100" Style="{StaticResource
DropDownStyle}"/>
```

While this is easy enough to set on one control, if you have several controls it can be tedious to set the style on each one. That's where the **ImplicitStyleManager** comes in. See [Using the ImplicitStyleManager](#) (page 29) for more information.

## Using the ImplicitStyleManager

The **ImplicitStyleManager** lets you set styles implicitly in Silverlight as you might in WPF. You can find the **ImplicitStyleManger** in the **System.Windows.Controls.Theming.Toolkit.dll** assembly installed with the Silverlight Toolkit.

To use the **ImplicitStyleManager** add a reference in your project to the **System.Windows.Controls.Theming.Toolkit.dll** assembly and add its namespace to the initial **UserControl** tag as in the following markup:

```
<UserControl
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:c1="clr-
namespace:C1.Silverlight;assembly=C1.Silverlight" xmlns:theming="clr-
namespace:System.Windows.Controls.Theming;assembly=System.Windows.Controls.Th
eming.Toolkit" x:Class="C1Theming.MainPage" Width="640" Height="480">
```

Once you've added the reference and namespace you can use the **ImplicitStyleManager** in your application. For example, in the following markup a style is added and implicitly implemented:

```
<Grid x:Name="LayoutRoot" Background="White"
theming:ImplicitStyleManager.ApplyMode="OneTime">
    <Grid.Resources>
        <Style TargetType="c1:C1DropDown">
            <Setter Property="Background" Value="Red" />
        </Style>
    </Grid.Resources>
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <c1:C1DropDown Margin="5" Content="C1DropDown" Height="30"
Width="100"/>
```

```
</StackPanel>
</Grid>
```

## Applying Themes to Controls

You can easily customize your application, by applying one of the built-in themes to your `ComponentOne Silverlight` control. Each of the built-in themes is based on a Silverlight Toolkit theme. For information about each of the built-in themes, see [Available Themes](#) (page 15). In this example, you'll add the `RainierOrange` theme to the `C1DropDown` control on a page.

To apply the theme, complete the following steps:

1. In Visual Studio, select **File | New Project**.
2. In the **New Project** dialog box, select the language in the left pane and in the right-pane select **Silverlight Application**. Enter a **Name** and **Location** for your project and click **OK**.
3. In the **New Silverlight Application** dialog box, leave the default settings and click **OK**.

A new application will be created and should open with the `MainPage.xaml` file displayed in XAML view.

4. Place the mouse cursor between the `<Grid>` and `</Grid>` tags in XAML view.

You will add the theme and control to the Grid in the next steps.

5. Navigate to the Visual Studio Toolbox and double-click on the **C1ThemeRainierOrange** icon to declare the theme. The theme's namespace will be added to the page and the theme's tags will be added to the Grid in XAML view. The markup will appear similar to the following:

```
<UserControl xmlns:my="clr-
namespace:C1.Silverlight.Theming.RainierOrange;assembly=C1.Silverlight.
Theming.RainierOrange" x:Class="C1Silverlight.MainPage"
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:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
  <Grid x:Name="LayoutRoot">
    <my:C1ThemeRainierOrange></my:C1ThemeRainierOrange>
  </Grid>
</UserControl>
```

Any controls that you add within the theme's tags will now be themed.

6. Place your cursor between the `<my:C1ThemeRainierOrange>` and `</my:C1ThemeRainierOrange>` tags.
7. In the Toolbox, double-click the **C1DropDown** icon to add the control to the project. The `C1.Silverlight` namespace will be added to the page and the control's tags will be added within the theme's tags in XAML view. The markup will appear similar to the following:

```
<UserControl xmlns:c1="clr-
namespace:C1.Silverlight;assembly=C1.Silverlight"
xmlns:my="clr-
namespace:C1.Silverlight.Theming.RainierOrange;assembly=C1.Silverlight.
Theming.RainierOrange" x:Class="C1Silverlight.MainPage"
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:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
```

```

    <Grid x:Name="LayoutRoot">
    <my:C1ThemeRainierOrange>
        <cl:C1DropDown Width="100" Height="30"></cl:C1DropDown>
    </my:C1ThemeRainierOrange>
    </Grid>
</UserControl>

```

### What You've Accomplished

Run your project and observe that the **C1DropDown** control now appears in the **RainierOrange** theme. Note that you can only set the **Content** property on the theme once, so to theme multiple controls using this method you will need to add a panel, for example a **Grid** or **StackPanel**, within the theme and then add multiple controls within the panel.

You can also use the **ImplicitStyleManager** to theme all controls of a particular type. For more information, see [Using the ImplicitStyleManager](#) (page 29).

## Applying Themes to an Application

The following topic details one method of applying a theme application-wide in Visual Studio. In this topic you'll add a class to your application that initializes a built-in theme. You'll then apply the theme to the **MainPage** of your application.

To apply the theme, complete the following steps:

1. In Visual Studio, select **File | New Project**.
2. In the **New Project** dialog box, select the language in the left pane and in the right-pane select **Silverlight Application**. Enter a **Name** and **Location** for your project and click **OK**.
3. In the **New Silverlight Application** dialog box, leave the default settings and click **OK**.

A new application will be created and should open with the **MainPage.xaml** file displayed in XAML view.

4. In the Solution Explorer, right-click the project and choose **Add Reference**.
5. In the **Add Reference** dialog box choose the **C1.Silverlight.Theming** and **C1.Silverlight.Theming.RainierOrange** assemblies and click **OK**.
6. In the Solution Explorer, right-click the project and select **Add | New Item**.
7. In the **Add New Item** dialog box, choose **Class** from the templates list, name the class "MyThemes", and click the **Add** button to create and a new class. The newly created **MyThemes** class will open.
8. Add the following import statements to the top of the class:

- Visual Basic
 

```
Imports C1.Silverlight.Theming
Imports C1.Silverlight.Theming.RainierOrange
```

- C#
 

```
using C1.Silverlight.Theming;
using C1.Silverlight.RainierOrange;
```

9. Add code to the class so it appears like the following:

- Visual Basic
 

```
Public Class MyThemes
    Private _myTheme As C1Theme = Nothing
    Public ReadOnly Property MyTheme() As C1Theme
        Get
            If _myTheme Is Nothing Then
                _myTheme = New C1ThemeRainierOrange()
            End If
        End Get
    End Property
End Class
```

```

        End If
        Return _myTheme
    End Get
End Property
End Class

```

- **C#**

```

public class MyThemes
{
    private static C1Theme _myTheme = null;
    public static C1Theme MyTheme
    {
        get
        {
            if (_myTheme == null)
                _myTheme = new C1ThemeRainierOrange();
            return _myTheme;
        }
    }
}

```

10. In the Solution Explorer, double-click the **App.xaml.vb** or **App.xaml.cs** file.

11. Add the following import statement to the top of the file, where *ProjectName* is the name of your application:

- Visual Basic

```
Imports ProjectName
```

- C#

```
using ProjectName;
```

12. Add code to the **Application\_Startup** event of the **App.xaml.vb** or **App.xaml.cs** file so it appears like the following:

- Visual Basic

```

Private Sub Application_Startup(ByVal o As Object, ByVal e As
StartupEventArgs) Handles Me.Startup
    Dim MyMainPage As New MainPage()
    Dim themes As New MyThemes
    themes.MyTheme.Apply(MyMainPage)
    Me.RootVisual = MyMainPage
End Sub

```

- C#

```

private void Application_Startup(object sender, StartupEventArgs e)
{
    MainPage MyMainPage = new MainPage();
    MyThemes.MyTheme.Apply(MyMainPage);
    this.RootVisual = MyMainPage;
}

```

Now any control you add to the **MainPage.xaml** file will automatically be themed.

13. Return to the **MainPage.xaml** file and place the mouse cursor between the `<Grid>` and `</Grid>` tags in XAML view.

14. In the Toolbox, double-click the **C1DropDown** icon to add the control to the project.

15. Update the control's markup so it appears like the following:

```
<c1:C1DropDown Width="100" Height="30"></c1:C1DropDown>
```

## What You've Accomplished

Run your project and observe that the **C1DropDown** control now appears in the **RainierOrange** theme. To change the theme chosen, now all you would need to do is change the theme in the **MyThemes** class.

For example, to change to the ExpressionDark theme:

1. Add a reference to the C1.Theming.Silverlight.ExpressionDark.dll assembly.
2. Open the **MyThemes** class in your project and add the following import statements to the top of the class:

- Visual Basic

```
Imports C1.Silverlight.Theming.ExpressionDark
```

- C#

```
using C1.Silverlight.Theming.ExpressionDark;
```

3. Update code in the class so it appears like the following:

- Visual Basic

```
Public Class MyThemes
    Private _myTheme As C1Theme = Nothing
    Public ReadOnly Property MyTheme() As C1Theme
        Get
            If _myTheme Is Nothing Then
                _myTheme = New C1ThemeExpressionDark()
            End If
            Return _myTheme
        End Get
    End Property
End Class
```

- C#

```
public class MyThemes
{
    private static C1Theme _myTheme = null;
    public static C1Theme MyTheme
    {
        get
        {
            if (_myTheme == null)
                _myTheme = new C1ThemeExpressionDark();
            return _myTheme;
        }
    }
}
```

Note that the above steps apply the theme to the **MainPage.xaml** file. To apply the theme to additional pages, you would need to add the following code to each page:

- Visual Basic

```
Dim themes As New MyThemes
themes.MyTheme.Apply(MyMainPage)
```

- C#

```
MyThemes.MyTheme.Apply(LayoutRoot);
```

The theme will then be applied to the page. So, you only have to change one line of code to the class to change the theme, and you only have to add one line of code to each page to apply the theme.

# 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.

## ClearStyle Properties

With each release, ComponentOne will be adding ClearStyle functionality to more controls. Currently several Silverlight and WPF controls support ClearStyle. The following table lists all of the ClearStyle-supported Silverlight controls as well as the ClearStyle properties that each supports.

Property	Supported Controls
AlternatingBackground	C1Scheduler
AppointmentForeground	C1Scheduler
AlternatingRowBackground	C1DataGrid
AlternatingRowForeground	C1DataGrid
Background	C1Accordion, C1AccordionItem, C1ColorPicker, C1ComboBox, C1ComboBoxItem, C1ContextMenu, C1CoverFlow, C1DataGrid, C1DateTimePicker, C1Docking, C1DropDown, C1Expander, C1ExpanderButton, C1FilePicker, C1HeaderedContentControl, C1Map, C1MediaPlayer, C1Menu, C1MenuItem, C1NumericBox, C1Window, C1RangeSlider, C1PropertyGrid, C1Scheduler, C1TabControl, C1TabItem, C1TextBoxBase, C1TimeEditor, C1ToolBar, C1ToolBarGroup, C1ToolBarStrip, C1ToolBarStripItem, C1TreeView, C1TreeViewItem, C1Window
ButtonBackground	C1ComboBox, C1CoverFlow, C1DropDown, C1FilePicker, C1NumericBox, C1TimeEditor, C1ToolBarStrip, C1Window

ButtonForeground	C1ComboBox, C1CoverFlow, C1DropDown, C1FilePicker, C1NumericBox, C1TimeEditor, C1ToolBarStrip, C1Window
CaretBrush	C1ColorPicker, C1ComboBox, C1DateTimePicker, C1NumericBox, C1TextBoxBase, C1TimeEditor
CategoryBackground	C1PropertyGrid
CategoryForeground	C1PropertyGrid
ControlBackground	C1Scheduler
ControlForeground	C1Scheduler
ExpandedBackground	C1AccordionItem, C1Expander, C1ExpanderButton,
FocusBrush	C1ColorPicker, C1ComboBox, C1DataGrid, C1DateTimePicker, C1DropDown, C1Expander, C1ExpanderButton, C1FilePicker, C1MediaPlayer, C1NumericBox, C1Window, C1RangeSlider, C1TextBoxBase, C1TimeEditor, C1ToolBar, C1ToolBarGroup, C1ToolBarStrip, C1ToolBarStripItem
Header	C1Accordion, C1AccordionItem, C1Expander, C1HeaderedContentControl, C1Window
HighlightedBackground	C1ContextMenu, C1Menu, C1MenuList, C1MenuItem
HorizontalGridLinesBrush	C1DataGrid
MouseOverBrush	C1Accordion, C1AccordionItem, C1ColorPicker, C1ComboBox, C1ComboBoxItem, C1CoverFlow, C1DataGrid, C1DateTimePicker, C1Docking, C1DropDown, C1Expander, C1ExpanderButton, C1FilePicker, C1Map, C1MediaPlayer, C1NumericBox, C1RangeSlider, C1PropertyGrid, C1TabControl, C1TabItem, C1TextBoxBase, C1TimeEditor, C1ToolBar, C1ToolBarGroup, C1ToolBarStrip, C1ToolBarStripItem, C1TreeView, C1TreeViewItem, C1Window
OpenedBackground	C1ContextMenu, C1Menu, C1MenuList, C1MenuItem
PressedBrush	C1ColorPicker, C1ComboBox, C1CoverFlow, C1DataGrid, C1DateTimePicker, C1DropDown, C1ExpanderButton, C1FilePicker, C1Map, C1MediaPlayer, C1NumericBox, C1PropertyGrid, C1RangeSlider, C1TextBoxBase, C1TimeEditor, C1ToolBar, C1ToolBarGroup, C1ToolBarStrip, C1ToolBarStripItem, C1Window
RowBackground	C1DataGrid
RowForeground	C1DataGrid
SelectedBackground	C1ComboBox, C1ComboBoxItem, C1DataGrid, C1Scheduler, C1TabControl, C1TabItem, C1TreeView, C1TreeViewItem,
SelectionBackground	C1ColorPicker, C1ComboBox, C1DateTimePicker, C1FilePicker, C1NumericBox, C1TextBoxBase, C1TimeEditor
SelectionForeground	C1ColorPicker, C1ComboBox, C1DateTimePicker, C1FilePicker, C1NumericBox, C1TextBoxBase, C1TimeEditor
TabItemBackground,	C1Docking
TabStripBackground	C1Docking, C1TabControl
TabStripForeground	C1Docking, C1TabControl
TodayBackground	C1Scheduler



# PDF for Silverlight

Create Adobe PDF documents from your apps using **ComponentOne PDF™ for Silverlight**. The commands provided for adding content to documents are similar to the ones available in the .NET Graphics class. PDF for Silverlight gives you security, compression, outlining, hyper-linking, and attachments.

One of the main features in the **C1PdfDocument** class is its ease of use. The commands provided for adding content to documents are similar to the ones available in the WinForms Graphics class. If you know how to display text and graphics in WinForms, then you already know how to use **C1PdfDocument**.

**C1PdfDocument** uses a Point-based coordinate system with the origin at the top-left corner of the page. This is similar to the default coordinate system used by .NET, but is different from the default PDF coordinate system (where the origin is on the bottom-left corner of the page).

**C1PdfDocument** supports many advanced features included in the PDF specification, including security, compression, outlining, hyper-linking, and file attachments.

The main limitation of the Silverlight version is that it only supports the Acrobat Reader built-in fonts: Times, Helvetica, Courier, and Symbol. This is because embedding other fonts types would require access to font outline information which is not available to Silverlight. In future versions, we may add support for downloading this information from the server. Acrobat Forms and text annotations are also not supported at this time.

## PDF for Silverlight Features

**ComponentOne PDF for Silverlight** supports most of the advanced features included in the PDF specification, including security, compression, outlining, hyperlinking, and attachments.

The following are some of the features of **PDF for Silverlight** that you may find useful:

- **Easily Add Content**

The C1PdfDocument class is easy to use. The commands provided for adding content to documents are similar to the ones available in the WinForms Graphics class. If you know how to display text and graphics in WinForms, you already know how to use C1PdfDocument in Silverlight. Add text, images, lines, rectangles, ellipses, pies, arcs, rounded rectangles, polygons, Bezier curves, and more.

- **Export Your UI**

Export your Silverlight UI "as-is" directly to PDF with selectable text and images. This is an experimental feature which handles most common UI elements in the visual tree. Or you can simply export the UI to PDF using bitmaps. Just point the C1PdfDocument to your root visual element.

- **Familiar Syntax Using DrawImage Method**

Adding images to PDF documents is easy; all the work is done by the DrawImage method. DrawImage draws a given image at a specified location and has parameters that provide control over the image alignment and scaling. You can render any regular .NET Image object, including metafiles.

- **Fast Rendering and Compression of Images**

PDF allows multiple levels of compression giving options for high quality and small file size. Metafiles are parsed and converted into vector graphics commands and thus retain the best possible resolution. If you want to add charts or technical drawings to your PDF document, metafiles are better than bitmap images. Add attachments to PDF files

- **Attachments**

Attachments can contain any kind of file, including spreadsheets with detailed information that would clutter the main document, multimedia files with movies and sound, sample code, and more. Adding an

attachment to your PDF file is easy. Simply specify which file you want to attach, what area of the page should contain the attachment, and optionally, the appearance of the attachment.

- **Security and Permissions**

If your PDF documents contain sensitive information, you can encrypt them so that only authorized users can access it. There is a separate password for the owner of the document and for all other users. The user's access can be selectively restricted to allow only certain operations, such as viewing, printing, or editing the document.

- **Outline Structure**

Most long PDF documents contain an outline structure that is displayed on a pane on the left of the reader. The outline makes it easy to browse through a document's structure and find specific topics. With PDF for Silverlight, you can build this outline structure by adding outline entries (bookmarks).

- **Hyperlinks and Local links**

PDF provides methods for adding hyperlinks and hyperlink targets to your PDF documents. You can also add local links, that when clicked take the user to another location within the same PDF document. This type of link is useful when you want to implement some type of cross-referencing within the document, such as a table of contents or an index.

- **Document Information and Viewer Preferences**

PDF allows you to add meta data to the PDF documents you create. Specify author, creation date, keywords, and so on. You can also provide default viewer preferences to be applied when the document is opened in the Adobe Reader. Specify the initial page layout, window position, as well as reader toolbar and menu visibility.

# PDF for Silverlight Quick Start

The following quick start guide is intended to get you up and running with **PDF for Silverlight**.

To create PDF documents using **C1PdfDocument**, three steps are required:

1. Create a **C1PdfDocument** object.
2. Add content to the document. This usually involves calling the **DrawString** method.
3. Save the document to a stream using the **Save** method.

In this quick start you will create a new project with a **C1PdfDocument** object, add content to the document, and save the document.

## Step 1 of 4: Creating an Application with the C1PdfDocument Object

In this step, you'll create a Silverlight application and add a **C1PdfDocument** object.

1. In Visual Studio, select **File | New | Project**.
2. In the **New Project** dialog box, select a language in the left pane, and in the templates list select **Silverlight Application**. Enter a **Name** for your project and click **OK**. The **New Silverlight Application** dialog box will appear.
3. Click **OK** to close the **New Silverlight Application** dialog box and create your project.
4. Right-click the project name in the Solution Explorer and select **Add Reference**.
5. In the **Add Reference** dialog box, locate and select the **C1.Silverlight.Pdf.dll** and click **OK** to add the reference to your project.

6. Open the **MainPage.xaml.cs** file in Visual Studio and add the following code at the top of the page with the **using** statements:

```
using C1.Silverlight.Pdf;
using System.IO;
```

7. Then add the following code for the MainPage:

```
// Create the C1PdfDocument object
C1PdfDocument pdf = new C1PdfDocument();

public MainPage()
{
    InitializeComponent();

    // Make the window hit test visible.
    LayoutRoot.Background = new SolidColorBrush(Colors.Transparent);
}
```

You've successfully created a Silverlight application with a **C1PdfDocument** object.

## Step 2 of 4: Adding Content to the Page

In this step you'll add some content to the document using the **DrawString** method.

In the **MainPage.xaml.cs** file immediately following the new **C1PdfDocument** object you created in step 1, add the following code:

```
protected override void OnMouseLeftButtonDown(MouseButtonEventArgs e)
{
    // Add content to the page
    Rect rc = pdf.PageRectangle;
    rc.X = rc.Y = 72;
    Font font = new Font("Arial", 12);
    pdf.DrawString("Hello World!", font, Colors.Black, rc);
}
```

## Step 3 of 4: Saving the document

In this step you'll save the document using the **Save** method.

In the **MainPage.xaml.cs** file immediately following the code you used to add content to the document in step 2, add the following code:

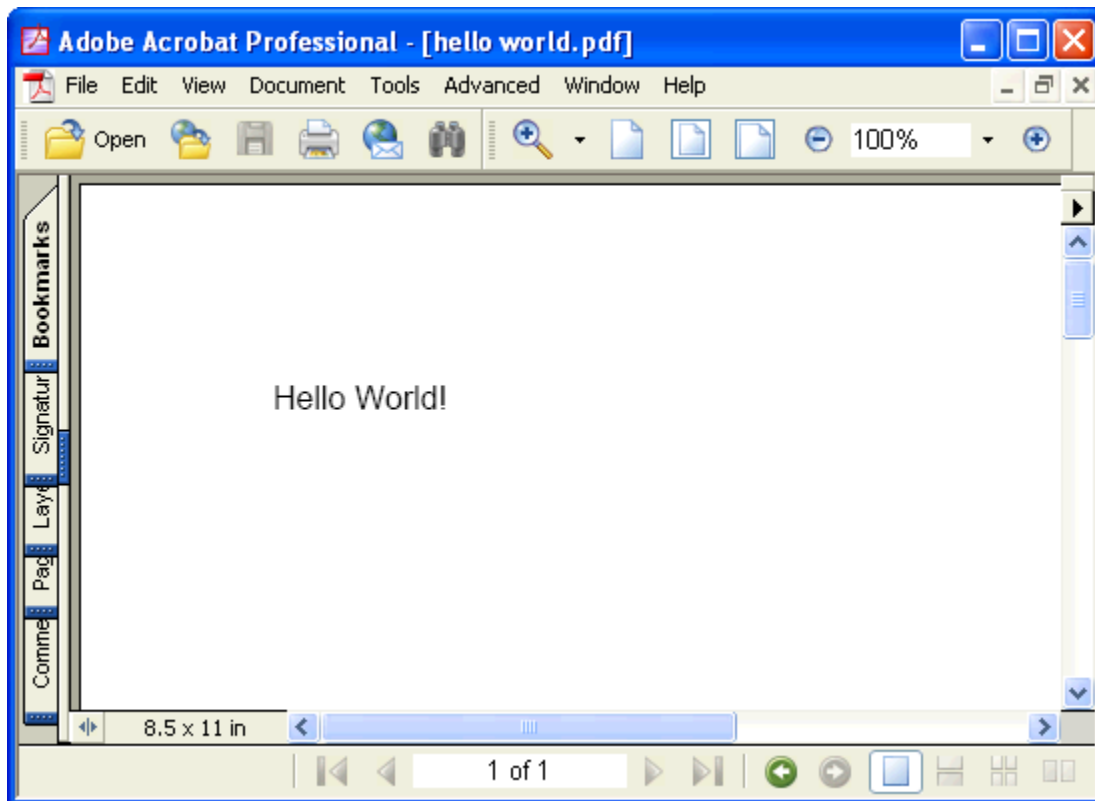
```
// Save the document
SaveFileDialog dlg = this.GetFileDialog();
if (dlg != null)
{
    using (Stream stream = dlg.OpenFile())
    {
        pdf.Save(stream);
    }
}

private SaveFileDialog GetFileDialog()
{
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.DefaultExt = ".pdf";
    bool? dr = dlg.ShowDialog();
    return ((dr.HasValue && dr.Value) ? dlg : null);
}
```

## Step 4 of 4: Running the Application

In this step, you will run the application and view the hello world.pdf document.

From the **Debug** menu, select **Start Debugging** to view the new PDF.



The code starts by retrieving the page rectangle, expressed in points. It then adds a one-inch margin around the page (72 points). Finally, the code creates a **Font** object and calls the **C1PdfDocument.DrawString** method to write "Hello World!" on the page. This is exactly what you would do if you were writing to a **Graphics** object in .NET and is what makes **PDF for Silverlight** so easy to use.

One important thing to remember is that **PDF for Silverlight** uses a point-based coordinate system with the origin at the top-left corner of the page. This is similar to the default coordinate system used by .NET, but is different from the default PDF coordinate system (where the origin is on the bottom-left corner of the page). In this example, the top left point of the "H" in "Hello World" is located at [1,1].

Because the coordinate system is based on points, rather than pixels, **PDF for Silverlight** uses **Rect**, **SizeF**, and **PointF** structures, which have members of type **float**, rather than **Rectangle**, **Size**, and **Point**, which have members of type **int**.

Congratulations! You have completed the **PDF for Silverlight** quick start tutorial.

# Using ComponentOne PDF for Silverlight

The following topics provide details on how to add text, images, graphics, pages and overlays, bookmarks and annotations, and security and permissions to **PDF for Silverlight** documents.

## Adding Text

The following topics provide information on drawing, measuring, and managing the flow of text.

### Drawing Text

Adding text to **PDF for Silverlight** documents is easy – all the work is done by the **C1PdfDocument.DrawString** method.

**C1PdfDocument.DrawString** draws a given string at a specified location using a given font and brush. For example:

- Visual Basic

```
pdf.DrawString("Hello World!", font, Colors.Black, rect)
```

- C#

```
pdf.DrawString("Hello World!", font, Colors.Black, rect);
```

By default, **C1PdfDocument.DrawString** will align the text to the left and to the top of the given rectangle, will wrap the string within the rectangle, and will not clip the output to the rectangle. You can change all these options by specifying a *StringFormat* parameter in the call to **C1PdfDocument.DrawString**. The **StringFormat** has members that allow you to specify the horizontal alignment (**Alignment**), vertical alignment (**LineAlignment**), and flags that control wrapping and clipping (**FormatFlags**).

For example, the code below creates a **StringFormat** object and uses it to align the text to the center of the rectangle, both vertically and horizontally:

- Visual Basic

```
Dim font As New Font("Arial", 12)
Dim rect As New Rect(72, 72, 100, 50)
Dim text As String = "Some long string to be rendered into a small
rectangle. "
text = text & text & text & text & text & text

' Center align string.
Dim sf As New StringFormat()
sf.Alignment = HorizontalAlignment.Center
sf.LineAlignment = VerticalAlignment.Center
pdf.DrawString(text, font, Colors.Black, rect, sf)
pdf.DrawRectangle(Pens.Gray, rect)
```

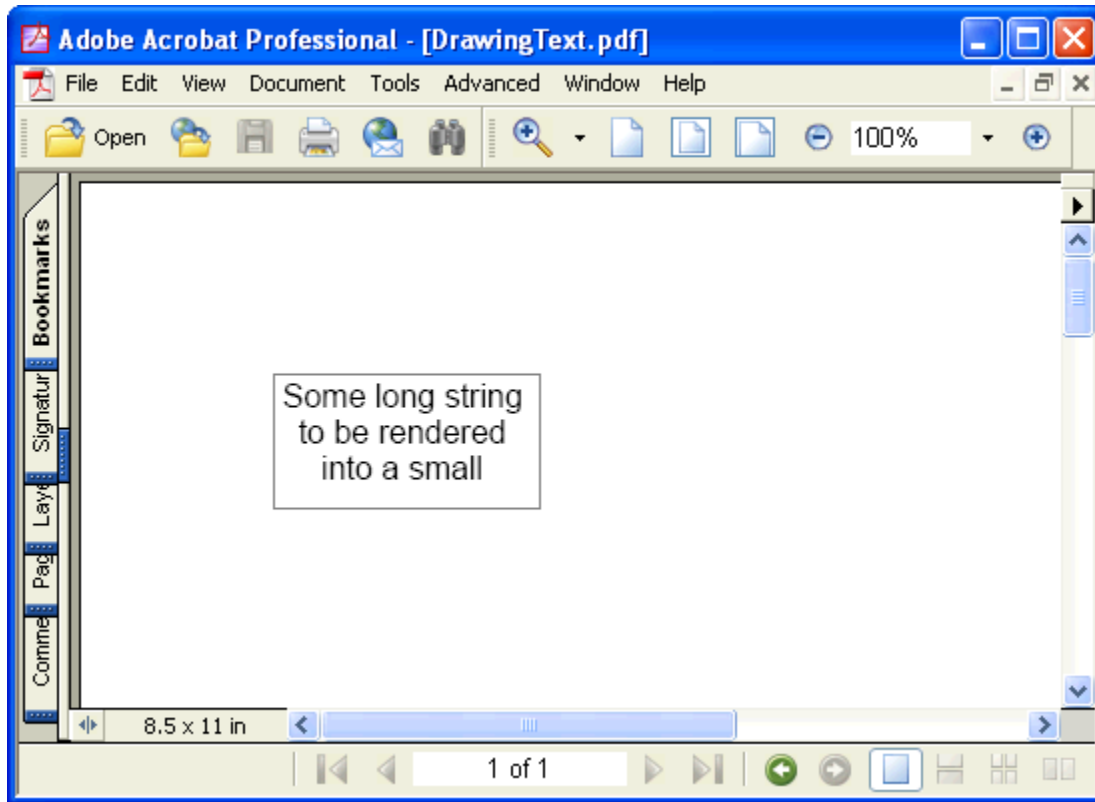
- C#

```
Font font = new Font("Arial", 12);
Rect rect = new Rect(72, 72, 100, 50);
string text = "Some long string to be rendered into a small rectangle. ";
text = text + text + text + text + text + text;

// Center align string.
StringFormat sf = new StringFormat();
sf.Alignment = HorizontalAlignment.Center;
```

```
sf.LineAlignment = VerticalAlignment.Center;
pdf.DrawString(text, font, Colors.Black, rect, sf);
pdf.DrawRectangle(Pens.Gray, rect);
```

Here is the resulting PDF document:



## Measuring Text

In many cases, you will need to check whether the string will fit on the page before you render it. You can use the **C1PdfDocument.MeasureString** method for that. **C1PdfDocument.MeasureString** returns a **SizeF** structure that contains the width and height of the string (in points) when rendered with a given font.

For example, the code below checks to see if a paragraph will fit on the current page and creates a page break if it has to. This will keep paragraphs together on a page:

- Visual Basic

```
Private Function RenderParagraph(text As String, font As Font, rect As Rect, rectPage As Rect) As Rect

    ' Calculate the necessary height.
    Dim sz As SizeF = _clpdf.MeasureString(text, font, rect.Width)
    rect.Height = sz.Height

    ' If it won't fit this page, do a page break.
    If rect.Bottom > rectPage.Bottom Then
        _clpdf.NewPage()
        rect.Y = rectPage.Top
    End If

    ' Draw the string.
```

```

    _clpdf.DrawString(text, font, Colors.Black, rect)

    ' Update rectangle for next time.
    Rect.Offset(0, rect.Height)
    Return rect
End Function

' Use the RenderParagraph method.
Dim font As New Font("Arial", 10)
Dim rectPage As Rect = _clpdf.PageRectangle()
rectPage.Inflate(-72, -72)
Dim rect As Rect = rectPage
Dim s As String
For Each s In myStringList
    rect = RenderParagraph(s, font, rect, rectPage)
Next s

```

- C#

```

private Rect RenderParagraph(string text, Font font, Rect rect, Rect
rectPage)
{
    // Calculate the necessary height.
    SizeF sz = _clpdf.MeasureString(text, font, rect.Width);
    rect.Height = sz.Height;

    // If it won't fit this page, do a page break.
    If (rect.Bottom > rectPage.Bottom)
    {
        _clpdf.NewPage();
        rect.Y = rectPage.Top;
    }

    // Draw the string.
    _clpdf.DrawString(text, font, Colors.Black, rect);

    // Update rectangle for next time.
    Rect.Offset(0, rect.Height);
    return rect;
}

// Use the RenderParagraph method.
Font font = new Font("Arial", 10);
Rect rectPage = _clpdf.PageRectangle();
rectPage.Inflate(-72, -72);
Rect rect = rectPage;
foreach (string s in myStringList)
{
    rect = RenderParagraph(s, font, rect, rectPage);
}

```

## Making Text Flow from Page to Page

The **C1PdfDocument.DrawString** method returns an integer. This is the index of the first character that was not printed because it did not fit the output rectangle. You can use this value make text flow from page to page or from one frame to another within a page. For example:

- Visual Basic

```

' Render a string spanning multiple pages.

```

```

While True

    ' Render as much as will fit into the rectangle.
    Dim nextChar As Integer
    nextChar = _clpdf.DrawString(text, font, Colors.Black, rectPage)

    ' Break when done.
    If nextChar >= text.Length Then
        Exit While
    End If

    ' Get rid of the part that was rendered.
    Text = text.Substring(nextChar)

    ' Move on to the next page.
    _clpdf.NewPage()
End While

```

- C#

```

// Render a string spanning multiple pages.
While (true)
{
    // Render as much as will fit into the rectangle.
    Int nextChar = _clpdf.DrawString(text, font, Colors.Black, rectPage);

    // Break when done.
    If (nextChar >= text.Length)
    {
        break;
    }

    // Get rid of the part that was rendered.
    Text = text.Substring(nextChar);

    // Move on to the next page.
    _clpdf.NewPage();
}

```

By combining the **C1PdfDocument.MeasureString** and **C1PdfDocument.DrawString** methods, you can develop rendering routines that provide extensive control over how paragraphs are rendered, including keeping paragraphs together on a page, keeping with the next paragraph, and controlling widows and orphans (single lines that render on the current or next page).

## Adding Images

Adding images to **PDF for Silverlight** documents is also easy, all the work is done by the **C1PdfDocument.DrawImage** method.

**C1PdfDocument.DrawImage** draws a given image at a specified location and has parameters that provide control over the image alignment and scaling. In the following example, this image is:

- Stretched to fill the given rectangle
- Center-aligned within the rectangle, scaled to keep the aspect ratio
- Aligned to the top-left corner of the rectangle, with the original size

This code is used to draw the same image three times:

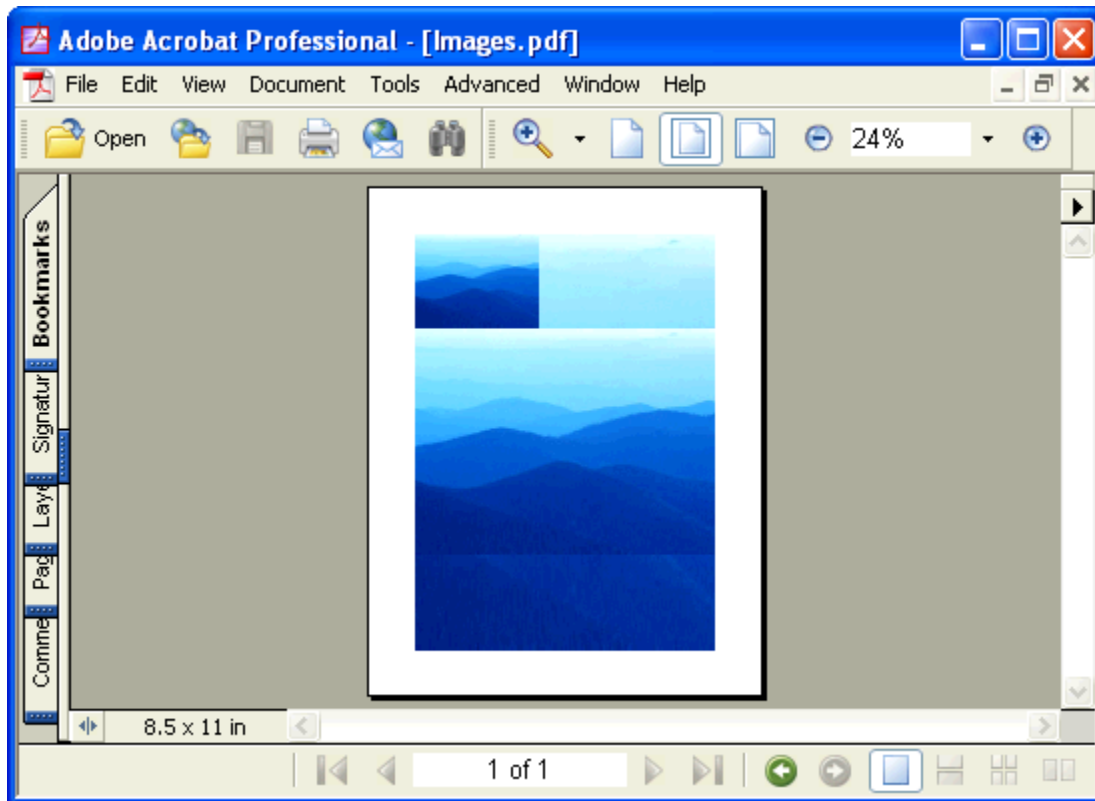
- Visual Basic

```
Dim rect As Rect = pdf.PageRectangle()  
rect.Inflate(- 72, - 72)  
  
' Stretch image to fill the rectangle.  
pdf.DrawImage(pictureBox1.Image, rect)  
  
' Center image within the rectangle, scale keeping aspect ratio.  
pdf.DrawImage(pictureBox1.Image, rect, ContentAlignment.MiddleCenter,  
C1.Silverlight.Pdf.ImageSizeModeEnum.Scale)  
  
' Render the image to the top left corner of the rectangle.  
pdf.DrawImage(pictureBox1.Image, rect, ContentAlignment.TopLeft,  
C1.Silverlight.Pdf.ImageSizeModeEnum.Clip)
```

- **C#**

```
Rect rect = pdf.PageRectangle();  
rect.Inflate(-72, -72);  
  
// Stretch image to fill the rectangle.  
pdf.DrawImage(pictureBox1.Image, rect);  
  
// Center image within the rectangle, scale keeping aspect ratio.  
pdf.DrawImage(pictureBox1.Image, rect, ContentAlignment.MiddleCenter,  
C1.Silverlight.Pdf.ImageSizeModeEnum.Scale);  
  
// Render the image to the top left corner of the rectangle.  
pdf.DrawImage(pictureBox1.Image, rect, ContentAlignment.TopLeft,  
1.C1Pdf.ImageSizeModeEnum.Clip);
```

The PDF document will look similar to this:



Notice that you can render any regular .NET Image object, including Metafiles. Metafiles are not converted into bitmaps; they are played into the document and thus retain the best possible resolution. If you want to add charts or technical drawings to your PDF document, Metafiles are better than bitmap images.

Bitmap images are managed automatically by **PDF for Silverlight**. If you render the same image several times (in a page header for example), only one copy of the image is saved into the PDF file.

## Adding Graphics

The **C1PdfDocument** class exposes several methods that allow you to add graphical elements to your documents, including lines, rectangles, ellipses, pies, arcs, rounded rectangles, polygons, Bezier curves, and so on.

The methods are a subset of those found in the .NET **Graphics** class, and use the same **Brush** and **Pen** classes to control the color and style of the lines and filled areas.

It is important to remember that **PDF for Silverlight** uses a coordinate system based on points, with the origin located at the top left of the page. (The default coordinate system for the .NET **Graphics** class is pixel-based.)

The example below illustrates how similar the graphics methods are between **PDF for Silverlight** and the .NET **Graphics** class. The sample declares a **C1PdfDocument** class called 'g' and calls methods to draw pies, splines, and other graphical elements.

The point of the sample is that if you replaced the **C1PdfDocument** class with a regular .NET **Graphics** object, you would be able to compile the code and get the same results:

- Visual Basic

```
' Create PDF document.
Dim g As New C1.Silverlight.Pdf.C1PdfDocument()

' Set up to draw.
Dim rect As New Rect(0, 0, 300, 200)
```

```

Dim text As String = "Hello world of .NET Graphics and PDF." +
ControlChars.Cr + ControlChars.Lf + "Nice to meet you."
Dim font As New Font("Times New Roman", 12, FontStyle.Italic Or
FontStyle.Underline)
Dim bezierPoints() As PointF = {New PointF(10F, 100F), New PointF(20F,
10F), New PointF(35F, 50F), New PointF(50F, 100F), New PointF(60F, 150F),
New PointF(65F, 100F), New PointF(50F, 50F)}

' Draw some pie slices.
Dim penWidth As Integer = 0
Dim penRGB As Integer = 0
g.FillPie(Colors.Red, rect, 0, 20F)
g.FillPie(Colors.Green, rect, 20F, 30F)
g.FillPie(Colors.Blue, rect, 60F, 12F)
g.FillPie(Colors.Gold, rect, - 80F, - 20F)

' Draw some arcs.
Dim startAngle As Single
For startAngle = 0 To 360 - 40 Step 40
    Dim penColor As Color = Color.FromArgb(penRGB, penRGB, penRGB)
    penWidth = penWidth + 1
    Dim pen As New Pen(penColor, penWidth)
    penRGB = penRGB + 20
    g.DrawArc(pen, rect, startAngle, 40F)
Next

' Draw a rectangle and some bezier splines.
g.DrawRectangle(Pens.Red, rect)
g.DrawBeziers(Pens.Blue, bezierPoints)
g.DrawString(text, font, Colors.Black, rect)

```

- **C#**

```

// Create PDF document.
Cl.Silverlight.Pdf.ClPdfDocument g = new
Cl.Silverlight.Pdf.ClPdfDocument();

// Set up to draw.
Rectangle rect = new Rect(0,0,300,200);
string text = "Hello world of .NET Graphics and PDF.\r\n" + "Nice to meet
you.";
Font font = new Font("Times New Roman", 12, FontStyle.Italic |
FontStyle.Underline);
PointF[] bezierPoints = new PointF[]
{
    new PointF(10f, 100f), new PointF(20f, 10f), new PointF(35f, 50f), new
PointF(50f, 100f), new PointF(60f, 150f), new PointF(65f, 100f), new
PointF(50f, 50f)
};

// Draw some pie slices.
int penWidth = 0;
int penRGB = 0;
g.FillPie(Colors.Red, rect, 0, 20f);
g.FillPie(Colors.Green, rect, 20f, 30f);
g.FillPie(Colors.Blue, rect, 60f, 12f);
g.FillPie(Colors.Gold, rect, -80f, -20f);

```

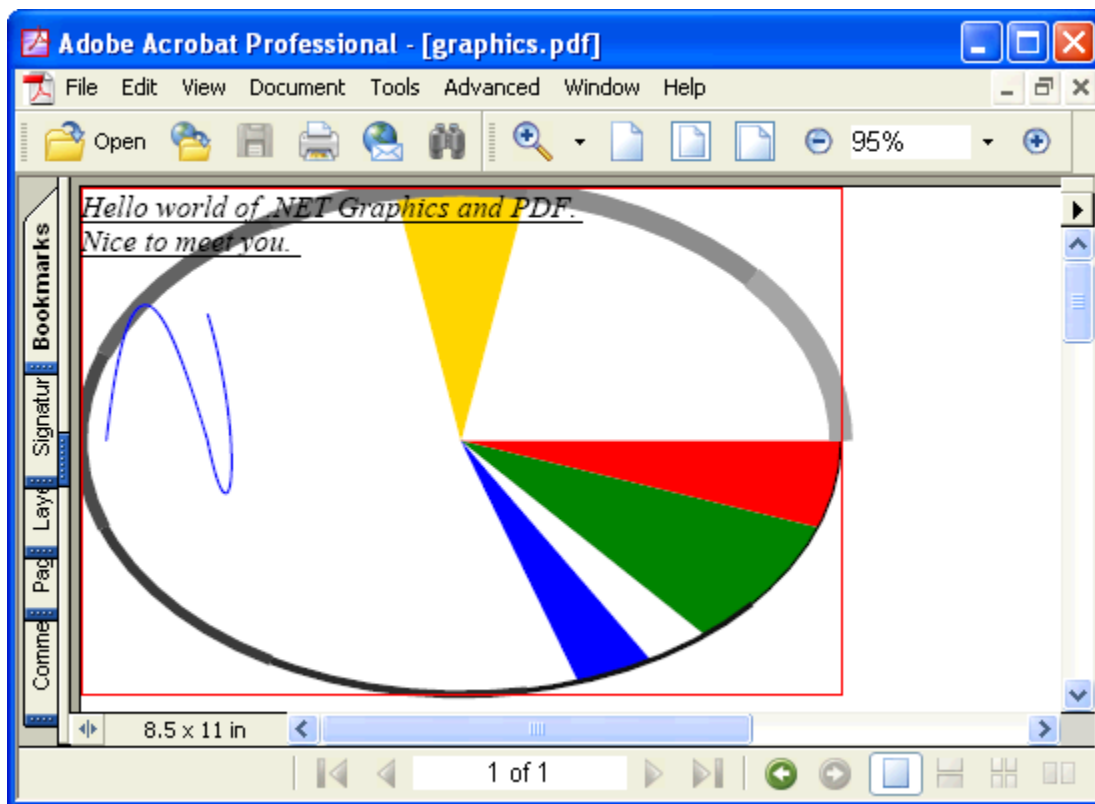
```

// Draw some arcs.
for (float startAngle = 0; startAngle < 360; startAngle += 40)
{
    Color penColor = Color.FromArgb(penRGB, penRGB, penRGB);
    Pen pen = new Pen(penColor, penWidth++);
    penRGB = penRGB + 20;
    g.DrawArc(pen, rect, startAngle, 40f);
}

// Draw a rectangle and some bezier splines.
g.DrawRectangle(Pens.Red, rect);
g.DrawBeziers(Pens.Blue, bezierPoints);
g.DrawString(text, font, Colors.Black, rect);

```

Here is the resulting PDF document:



## Creating Pages and Overlays

You may have noticed that in the previous examples, we started adding content to the document right after creating the **C1PdfDocument** object. This is possible because when you create the **C1PdfDocument**, it automatically adds an empty page to the document, ready to receive any type of content.

When you are done filling up the first page, you can add a new one using the **NewPage** method.

By default, all pages in the document have the same size and orientation. These parameters can be specified in the **C1PdfDocument** constructor. You can also change the page size and orientation at any time by setting the **C1PdfDocument.PaperKind**, **C1PdfDocument.PageSize**, and **C1PdfDocument.Landscape** properties. For example, the code below creates a document with all paper sizes defined by the **PaperKind** enumeration:

- Visual Basic

```
Dim font As New Font("Arial", 9)
Dim sf As New StringFormat()
sf.Alignment = HorizontalAlignment.Center
sf.LineAlignment = VerticalAlignment.Center

' Create one page with each paper size.
Dim firstPage As Boolean = True
Dim pk As PaperKind
For Each pk In System.Enum.GetValues(GetType(PaperKind))

    ' Skip custom size.
    If pk = PaperKind.Custom Then
        GoTo ContinueForEach1
    End If

    ' Add new page for every page after the first one.
    If Not firstPage Then
        _clpdf.NewPage()
    End If
    firstPage = False

    ' Set paper kind.
    _clpdf.PaperKind = pk

    ' Draw some content on the page.
    _clpdf.DrawString("PaperKind: " + pk.ToString(), font, Colors.Black,
        _clpdf.PageRectangle(), sf)

    ContinueForEach1:
Next pk
```

- C#

```
Font font = new Font("Arial", 9);
StringFormat sf = new StringFormat();
sf.Alignment = HorizontalAlignment.Center;
sf.LineAlignment = VerticalAlignment.Center;

// Create one page with each paper size.
bool firstPage = true;
foreach (PaperKind pk in Enum.GetValues(typeof(PaperKind)))
{
    // Skip custom size.
    if (pk == PaperKind.Custom)
    {
        continue;
    }

    // Add new page for every page after the first one.
    if (!firstPage)
    {
        _clpdf.NewPage();
    }
    firstPage = false;

    // Set paper kind.
```

```

    _clpdf.PaperKind = pk;

    // Draw some content on the page.
    _clpdf.DrawString("PaperKind: " + pk.ToString(), font, Colors.Black,
        _clpdf.PageRectangle(), sf);
}

```

You are not restricted to writing on the last page that was added to the document. You can use the **CIPdfDocument.CurrentPage** property to select which page you want to write to, and then use the regular drawing commands as usual. This is useful for adding content to pages after you are done rendering a document. For example, the code below adds footers to each page containing the current page number and the total of pages in the document (page n of m):

- Visual Basic

```

Private Sub AddFooters()
    Dim font As New Font("Helvetica", 7, FontStyle.Bold)
    Dim sf As New StringFormat()
    sf.Alignment = HorizontalAlignment.Center
    Dim page As Integer
    For page = 0 To clpdf.Pages.Count - 1

        ' Select page.
        _clpdf.CurrentPage = page

        ' Build rectangle for rendering the footer.
        Dim rect As Rect = _clpdf.PageRectangle()
        rect.Y = rect.Bottom

        ' Write the footer.
        Dim text As String
        text = String.Format("Page {0} of {1}", page + 1,
            _clpdf.Pages.Count)
        _clpdf.DrawString(text, font, Colors.Gray, rect, sf)
    Next page
End Sub

```

- C#

```

private void AddFooters()
{
    Font font = new Font("Helvetica", 7, FontStyle.Bold);
    StringFormat sf = new StringFormat();
    sf.Alignment = HorizontalAlignment.Center;
    for (int page = 0; page < _clpdf.Pages.Count; page++)
    {
        // Select page.
        _clpdf.CurrentPage = page;

        // Build rectangle for rendering the footer.
        Rect rect = _clpdf.PageRectangle();
        rect.Y = rect.Bottom - 36;

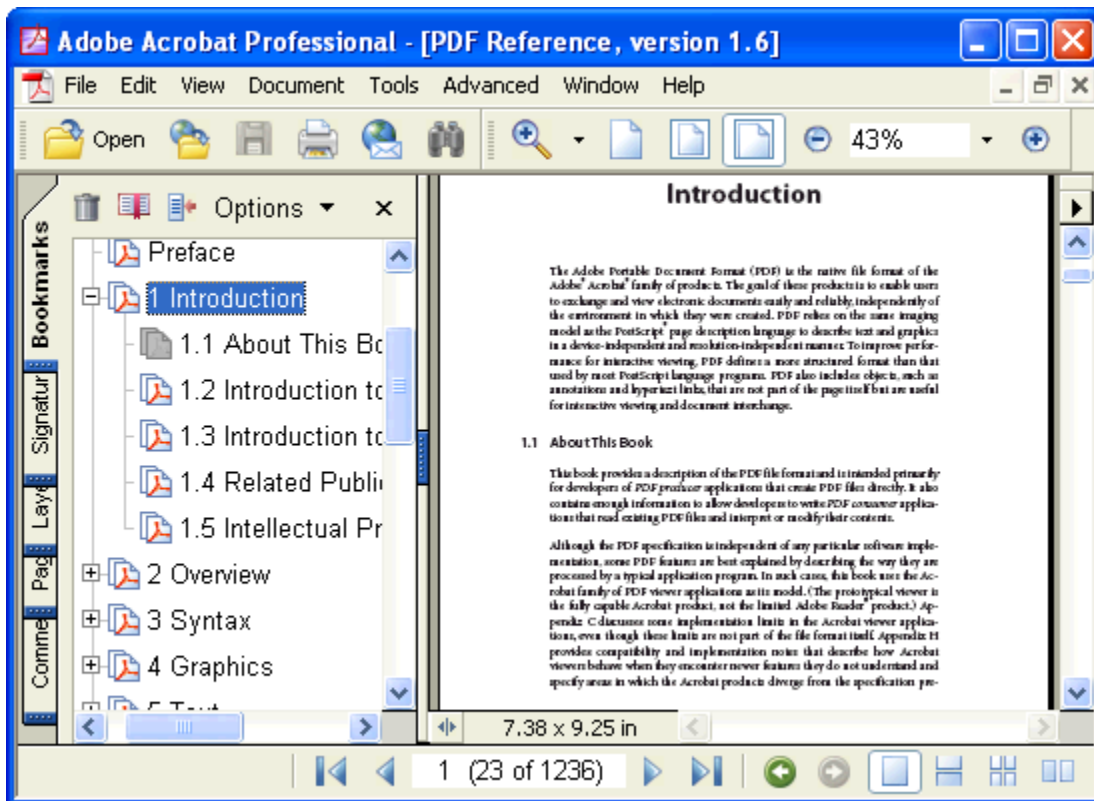
        // Write the footer.
        string text = string.Format("Page {0} of {1}", page+1,
            _clpdf.Pages.Count);
        _clpdf.DrawString(text, font, Colors.Gray, rect, sf);
    }
}

```

Note that the code uses the Pages property to get the page count. **Pages** is a collection based on the **ArrayList** class, and has methods that allow you to count and enumerate pages, as well as add and remove pages at specific positions. You can use the **Pages** collection to remove pages from certain locations in the document and re-insert them elsewhere.

## Adding Bookmarks to a PDF Document

When you open a PDF document using Adobe's Acrobat Reader application, you will notice that most long documents contain an outline structure that is displayed on a pane on the left of the reader. The outline makes it easy to browse through a document's structure and find specific topics. The picture below shows a PDF document with an outline:



The outline entries are called Bookmarks, and you can add them to your **PDF for Silverlight** documents using the **C1PdfDocument.AddBookmark** method. The **C1PdfDocument.AddBookmark** method takes three parameters: the title of the outline entry, the outline level, and the 'y' position of the entry on the current page (measured in points from the top of the page).

For example, the routine below adds a paragraph to a document and optionally marks it as a level-zero outline entry:

- Visual Basic

```
Private Function RenderParagraph(text As String, font As Font, rect As Rect, rectPage As Rect, outline As Boolean) As Rect

    ' If it doesn't fit on this page, add a page break.
    rect.Height = _clpdf.MeasureString(text, font, rect.Width).Height
    If rect.Bottom > rectPage.Bottom Then
        _clpdf.NewPage ()
    End If
End Function
```

```

        rect.Y = rectPage.Top
    End If

    ' Draw the string.
    _clpdf.DrawString(text, font, Colors.Black, rect)

    ' Add headings to outline.
    If outline Then
        _clpdf.DrawLine(Pens.Black, rect.X, rect.Y, rect.Right, rect.Y)
        _clpdf.AddBookmark(text, 0, rect.Y)
    End If

    ' Update rectangle for next time.
    rect.Offset(0, rect.Height)
    Return rect
End Function

```

- C#

```

private Rect RenderParagraph(string text, Font font, Rect rect, Rect
rectPage, bool outline)
{
    // If it doesn't fit on this page, add a page break.
    rect.Height = _clpdf.MeasureString(text, font, rect.Width).Height;
    if (rect.Bottom > rectPage.Bottom)
    {
        _clpdf.NewPage();
        rect.Y = rectPage.Top;
    }

    // Draw the string.
    _clpdf.DrawString(text, font, Colors.Black, rect);

    // Add headings to outline.
    if (outline)
    {
        _clpdf.DrawLine(Pens.Black, rect.X, rect.Y, rect.Right, rect.Y);
        _clpdf.AddBookmark(text, 0, rect.Y);
    }

    // Update rectangle for next time.
    rect.Offset(0, rect.Height);
    return rect;
}

```

## Adding Links to a PDF Document

The PDF specification allows you to add several types of annotations to your documents. Annotations are often added by hand, as highlights and notes. But they can also be added programmatically. C1PdfDocument provides methods for adding hyperlinks, hyperlink targets, and file attachments to your PDF documents.

To add a hyperlink to your document, use the AddLink method. AddLink method takes two parameters: a string that specifies a *url* and a *Rect* that specifies the area on the current page that should behave as a link.

Note that the AddLink method does not add any visible content to the page, so you will usually need another command along with AddLink to specify some text or an image that the user can see. For example, the code below adds a string that says "Visit ComponentOne" and a link that takes the user to the ComponentOne home page:

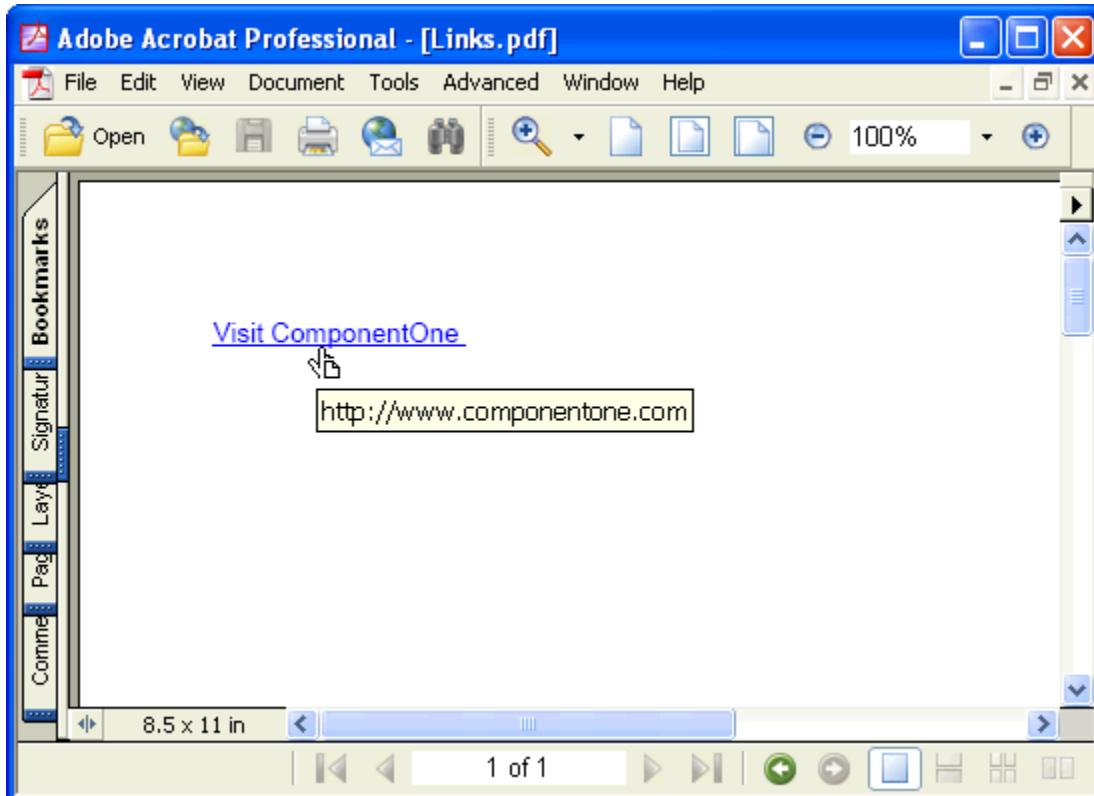
- Visual Basic

```
Dim rect As New Rect(50, 50, 100, 15)
Dim font As New Font("Arial", 10, FontStyle.Underline)
_clpdf.AddLink("http://www.componentone.com", rect)
_clpdf.DrawString("Visit ComponentOne", font, Colors.Blue, rect)
```

- C#

```
Rect rect = new Rect(50, 50, 100, 15);
Font font = new Font("Arial", 10, FontStyle.Underline);
_clpdf.AddLink("http://www.componentone.com", rect);
_clpdf.DrawString("Visit ComponentOne", font, Colors.Blue, rect);
```

Here is the resulting PDF document:



You can also add local links, which when clicked take the user to another location within the same PDF document. This type of link is useful when you want to implement some type of cross-referencing within the document, such as a table of contents or an index.

Local links are identical to regular hyperlinks, except for two things:

- The *url* parameter must start with a "#".
- You must specify the target location for the link using the *AddTarget* method. The *AddTarget* method takes the same parameters as *AddLink*, a string that specifies the name of the target and a rectangle that marks the area on the page that will be displayed when the user selects the link.

## Attaching Files to a PDF Document

Adding file attachments to PDF files is often a useful feature. Attachments can contain any kind of file, including spreadsheets with detailed information that would clutter the main document, multimedia files with movies and sound, sample code, and so on.

Adding file attachments to your **PDF for Silverlight** documents is easy. All you have to do is call the **C1PdfDocument.AddAttachment** method and specify which file you want to attach, what area of the page should contain the attachment, and optionally, the appearance of the attachment.

For example, the following code attaches all files in the application directory to the PDF document:

- Visual Basic

```
Dim rect As New Rect(100, 100, 60, 10)
Dim font As New Font("Arial", 9)

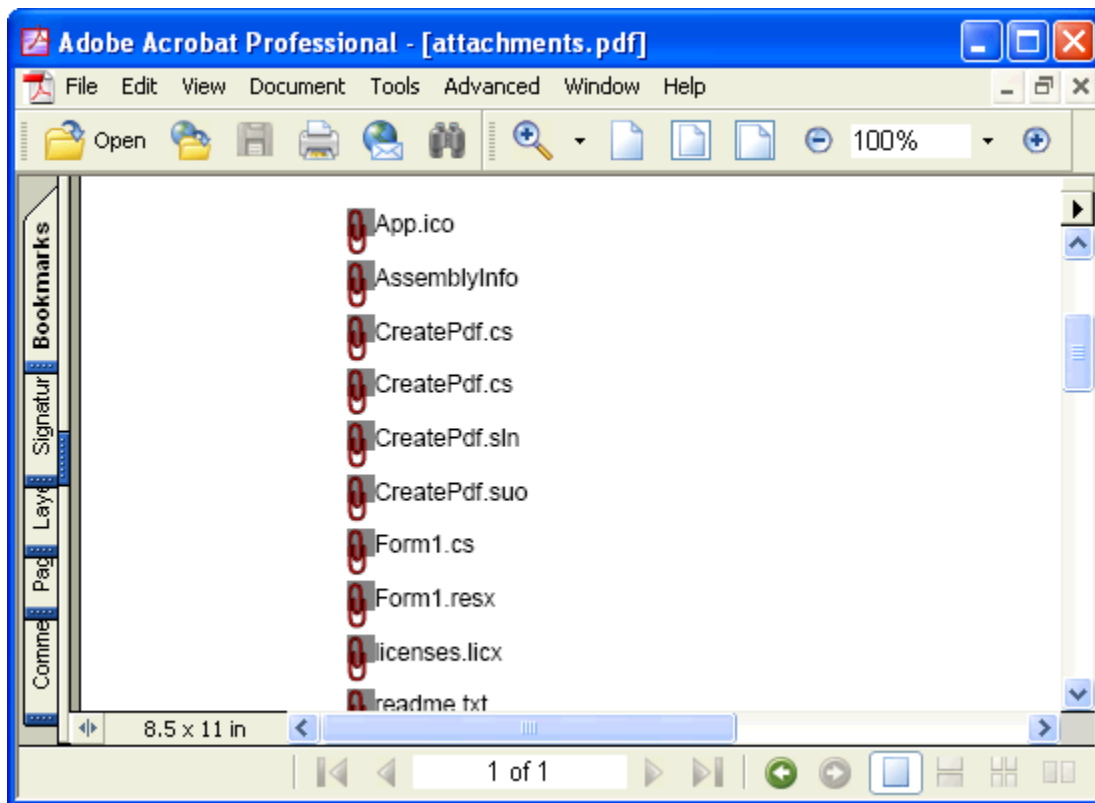
' Attach some files.
Dim path As String = "c:\temp\files"
Dim file As String
For Each file In Directory.GetFiles(path)
    Dim width As Single = rect.Width
    rect.Width = rect.Height
    _clpdf.FillRectangle(Colors.Gray, rect)
    _clpdf.AddAttachment(file, rect)
    rect.Width = width
    rect.X += rect.Height
    _clpdf.DrawString(Path.GetFileName(file), font, Colors.Black, rect)
    rect.X -= rect.Height
    rect.Y += 2 * rect.Height
Next file
```

- C#

```
Rect rect = new Rect(100, 100, 60, 10);
Font font = new Font("Arial", 9);

// Attach some files.
string path = @"c:\temp\files";
string[] files = Directory.GetFiles(path);
foreach (string file in files)
{
    float width = rect.Width;
    rect.Width = rect.Height;
    _clpdf.FillRectangle(Colors.Gray, rect);
    _clpdf.AddAttachment(file, rect);
    rect.Width = width;
    rect.X += rect.Height;
    _clpdf.DrawString(Path.GetFileName(file), font, Colors.Black, rect);
    rect.X -= rect.Height;
    rect.Y += 2 * rect.Height;
}
```

Here's what the PDF document looks like in Adobe's Acrobat Reader:



The attachments are displayed as icons (you can select from four predefined icons in the **AttachmentIconEnum** enumeration and you can also select the icon color). When the user moves the mouse over the attachment, the file name is displayed and the cursor changes to indicate there is an attachment available. The user can then right-click the attachment name to open the attachment or save it.

## Applying Security and Permissions

By default, anyone can open, copy, print, and edit PDF files. If your PDF documents contain sensitive information, however, you can encrypt them so that only authorized users can access it.

There is a separate password for the owner of the document and for all other users. The user's access can be selectively restricted to allow only certain operations, such as viewing, printing, or editing the document.

**PDF for Silverlight** provides a Security property that returns a **PdfSecurity** object. This object has properties that allow you to specify the **owner password** (required to change passwords and permissions for the document) and the **user password** (required to open the document). Additionally, the **PdfSecurity** object allows you to specify what permissions a regular user should have. For example you may allow users to see the document but not to print or edit it.

To use the **PDF for Silverlight** security features, simply set the passwords before you save the document. For example:

- Visual Basic

```
' Create the document as usual.
CreateDoc()

' Set passwords.
_clpdf.Security.OwnerPassword = "2mds%dfgd"
_clpdf.Security.UserPassword = "anyone"
_clpdf.Security.AllowEditAnnotations = False
```

```
_clpdf.Security.AllowEditContent = False
_clpdf.Security.AllowPrint = False
```

- C#

```
// Create the document as usual.
CreateDoc();

// Set passwords.
_clpdf.Security.OwnerPassword = "2mds%dfg";
_clpdf.Security.UserPassword = "anyone";
_clpdf.Security.AllowEditAnnotations = false;
_clpdf.Security.AllowEditContent = false;
_clpdf.Security.AllowPrint = false;
```

Save the document using the **Save** method as explained in the quick start topic, [Step 3 of 4: Saving the document](#) (page 39).

Note that you can specify permissions and set only the owner password, leaving the user password empty. In this case, anyone will be allowed to open the document, but only the owner will be allowed to change the permissions.

Note also that the encryption scheme used by **PDF for Silverlight** is public and is not 100% secure. There are ways to crack encrypted PDF documents. The security provided is adequate to protect your documents from most casual attacks, but if your data is truly sensitive you should not rely on PDF encryption alone.

## Using Metafiles

**PDF for Silverlight** makes it very easy to create documents, mainly because the object model mimics the well-known .NET **Graphics** model. However, not all methods available in the **Graphics** class are available in **PDF for Silverlight**. Plus, you may have existing code that draws to a **Graphics** object and that you do not want to rewrite even if most methods are very similar.

In these cases, you can reuse your existing .NET code by sending the drawing commands to a **Metafile**, then rendering the **Metafile** into **PDF for Silverlight** using the **C1PdfDocument.DrawImage** command. This method allows you to expose any graphics you create as images or as PDF documents.

For example, suppose you have an application that generates documents using the **PrintDocument** pattern of drawing each page into a **Graphics** object. You could then use the same methods to create a collection of metafiles, one per page, and then convert the list into a PDF document using the following code:

- Visual Basic

```
' Get the document as a list of Metafiles, one per page.
Dim pages As ArrayList = GetMetafiles()

' Loop through the pages and create a PDF document.
_clpdf.Clear()
Dim i As Integer
for i = 0 to pages.Count - 1

    ' Get ith page.
    Dim page As Metafile = CType(Metafile.FromFile(pages[i]), Metafile)
    If Not (page Is Nothing) Then

        ' Calculate the page size.
        Dim sz As.SizeF = page.PhysicalDimension
        sz.Width = Math.Round(sz.Width * 72.0F / 2540.0F, 2)
        sz.Height = Math.Round(sz.Height * 72.0F / 2540.0F, 2)

        ' Add a page and set the size.
        If i > 0 Then
```

```

        _clpdf.NewPage()
    End If
    _clpdf.PageSize = sz

    ' Draw the page into the PDF document.
    _clpdf.DrawImage(page, _clpdf.PageRectangle())
End If
Next

```

- C#

```

// Get the document as a list of Metafiles, one per page.
ArrayList pages = GetMetafiles();

// Loop through the pages and create a PDF document.
_clpdf.Clear();
for (int i = 0; i < pages.Count; i++)
{

    // Get ith page.
    Metafile page = (Metafile)Metafile.FromFile(pages[i]);
    if (page == null)
    {
        continue;
    }

    // Calculate the page size.
   .SizeF sz = page.PhysicalDimension;
    sz.Width = (float)Math.Round(sz.Width * 72f / 2540f, 2);
    sz.Height = (float)Math.Round(sz.Height * 72f / 2540f, 2);

    // Add a page and set the size.
    if (i > 0) _clpdf.NewPage();
    _clpdf.PageSize = sz;

    // Draw the page into the PDF document.
    _clpdf.DrawImage(page, _clpdf.PageRectangle());
}

```

Save the document using the **Save** method as explained in the quick start topic, [Step 3 of 4: Saving the document](#) (page 39).

The code gets each metafile on the list, calculates its size in points (each page could have a different size), then draws the metafile into the page. The metafiles could be generated by a reporting engine, drawing or charting program, or any application that can create metafile images.