
ComponentOne

ReportViewer for ASP.NET

Wijmo

Copyright © 2012 ComponentOne LLC. All rights reserved.

Corporate Headquarters

ComponentOne LLC

201 South Highland Avenue

3rd Floor

Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com

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

Sales

E-mail: sales@componentone.com

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

Trademarks

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

Warranty

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

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

Copying and Distribution

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

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

Table of Contents

ComponentOne ReportViewer for ASP.NET Wijmo Overview	1
Installing ComponentOne ReportViewer for ASP.NET Wijmo	1
Studio for ASP.NET Wijmo Setup Files.....	1
System Requirements	2
Uninstalling Studio for ASP.NET Wijmo.....	2
Deploying your Application in a Medium Trust Environment	3
End-User License Agreement	6
Licensing FAQs	6
What is Licensing?.....	6
How does Licensing Work?.....	7
Common Scenarios	7
Troubleshooting.....	9
Technical Support	11
Redistributable Files.....	12
Namespaces.....	13
Creating an ASP.NET Project	13
Adding the ReportViewer for ASP.NET Wijmo Components to a Project	15
Key Features.....	15
ReportViewer for ASP.NET Wijmo Quick Start	19
Step 1 of 3: Adding C1ReportViewer to the Page.....	19
Step 2 of 3: Adding Reports to the Control.....	19
Step 3 of 3: Running the Application.....	20
Wijmo Top Tips.....	23
The C1ReportViewer Control.....	23
Displaying Reports and Documents	23
Note on C1Report/C1PrintDocument Licensing.....	24
C1ReportViewer Elements	25
C1ReportViewer Toolbar.....	25
C1ReportViewer Navigation Panes	26

C1ReportViewer Preview Pane.....	28
Design-Time Support.....	29
C1ReportViewer Smart Tag.....	29
C1ReportViewer Context Menu.....	30
Run-Time Interaction.....	31
Printing a Report.....	31
Changing Report Flow.....	33
Zooming a Report.....	33
Navigating a Report.....	34
Searching a Report.....	35
ReportViewer for ASP.NET Wijmo Appearance.....	36
Themes.....	36
Changing Theme.....	39

ComponentOne ReportViewer for ASP.NET Wijmo Overview

Packed with rich features like zooming, paging, thumbnails, outlines, and more, ComponentOne WebPaper technology unleashes the capabilities of reporting in ASP.NET.

ComponentOne ReportViewer™ for ASP.NET Wijmo uses this technology to create pixel perfect reports in a high fidelity graphical form that are truly in a league of their own. Enjoy built-in toolbars for navigation, searching, printing, and exporting reports, and easy integration with other reporting formats including Crystal Reports, Microsoft Access, and Microsoft SQL Reporting Services. Turn any of your Web reports into powerful, printable, Web-ready documents in minutes.

For information about using **C1Report** and **C1PrintDocument**, see the [Reports for WinForms documentation](#).

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



Getting Started

Get started with the following topics:

- [Key Features](#) (page 15)
- [Quick Start](#) (page 19)
- [Control Elements](#) (page 25)

Installing ComponentOne ReportViewer for ASP.NET Wijmo

The following sections provide helpful information on installing **ComponentOne ReportViewer for ASP.NET Wijmo**.

C1ReportViewer Upgrade Note: You must remove the **reportService.ashx** file from your project before upgrading **C1ReportViewer** to a newer version. A new version of this file will be automatically created when the control is initialized for the first time.

Studio for ASP.NET Wijmo Setup Files

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

Bin	Contains copies of all binaries (DLLs, Exes) in the ComponentOne Visual Studio ASP.NET Wijmo package.
Wijmo	Contains files (at least a readme.txt) related to the product.

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

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

documentation for all Studio components.

Samples

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

Windows XP path: C:\Documents and Settings\<>username>\My Documents\ComponentOne Samples

Windows 7/Vista path: C:\Users\<>username>\Documents\ComponentOne Samples

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

Common	Contains support and data files that are used by many of the demo programs.
Studio for ASP.NET Wijmo	Contains a readme.txt file and the folders that make up the Control Explorer and other samples.

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

C1ReportViewer Upgrade Note: You must remove the **reportService.ashx** file from your project before upgrading **C1ReportViewer** to a newer version. A new version of this file will be automatically created when the control is initialized for the first time.

System Requirements

System requirements for **ComponentOne Studio for ASP.NET Wijmo** components include the following:

Operating Systems:	Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista™ Windows 7
Web Server:	Microsoft Internet Information Services (IIS) 6.0 or later
Environments:	.NET Framework 3.0 or later Visual Studio 2008 or later Internet Explorer 6.0 or later Firefox® 2.0 or later Safari® 2.0 or later

Uninstalling Studio for ASP.NET Wijmo

To uninstall Studio for ASP.NET Wijmo:

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

To uninstall **Studio for ASP.NET Wijmo** integrated help:

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

Deploying your Application in a Medium Trust Environment

Depending on your hosting choice, you may need to deploy your Web site or application in a medium trust environment. Often in a shared hosting environment, medium trust is required. In a medium trust environment several permissions are unavailable or limited, including OleDbPermission, ReflectionPermission, and FileIOPermission. You can configure your Web.config file to enable these permissions.

Note: ComponentOne controls will not work in an environment where reflection is not allowed.

ComponentOne ASP.NET Wijmo controls include the AllowPartiallyTrustedCallers() assembly attribute and will work under the medium trust level with some changes to the Web.config file. Since this requires some control over the Web.config file, please check with your particular host to determine if they can provide the rights to override these security settings.

Modifying or Editing the Config File

In order to add permissions, you can edit the existing web_mediumtrust.config file or create a custom policy file based on the medium trust policy. If you modify the existing web_mediumtrust.config file, all Web applications will have the same permissions with the permissions you have added. If you want applications to have different permissions, you can instead create a custom policy based on medium trust.

Edit the Config File

In order to add permissions, you can edit the existing web_mediumtrust.config file. To edit the existing web_mediumtrust.config file, complete the following steps:

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Open the web_mediumtrust.config file.
3. Add the permissions that you want to grant. For examples, see [Adding Permissions](#).

Create a Custom Policy Based on Medium Trust

In order to add permissions, you can create a custom policy file based on the medium trust policy. To create a custom policy file, complete the following steps:

1. Locate the medium trust policy file web_mediumtrust.config located by default in the %windir%\Microsoft.NET\Framework\{Version}\CONFIG directory.
2. Copy the web_mediumtrust.config file and create a new policy file in the same directory.
3. Give the new a name that indicates that it is your variation of medium trust; for example, AllowReflection_Web_MediumTrust.config.
4. Add the permissions that you want to grant. For examples, see [Adding Permissions](#).
5. Enable the custom policy file on your application by modifying the following lines in your web.config file under the <system.web> node:

```
<system.web>
  <trust level="CustomMedium" originUrl="" />

  <securityPolicy>
    <trustLevel name="CustomMedium"
policyFile="AllowReflection_Web_MediumTrust.config"/>
```

```
        </securityPolicy>
        ...
</system.web>
```

Note: Your host may not allow trust level overrides. Please check with your host to see if you have these rights.

Allowing Deserialization

To allow the deserialization of the license added to App_Licenses.dll by the Microsoft IDE, you should add the `SerializationFormatter` flag to security permission to the Web.config file. Complete the steps in the [Modifying or Editing the Config File](#) topic to create or modify a policy file before completing the following.

Add the `SerializationFormatter` flag to the `<IPermission class="SecurityPermission">` tag so that it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet
    class="NamedPermissionSet"
    version="1"
    Name="ASP.Net">
    <IPermission
      class="SecurityPermission"
      version="1"
      Flags="Assertion, Execution, ControlThread,
ControlPrincipal, RemotingConfiguration, SerializationFormatter"/>
    ...
  </PermissionSet>
</NamedPermissionSets>
```

Adding Permissions

You can add permission, including `ReflectionPermission`, `OleDbPermission`, and `FileIOPermission`, to the web.config file. Note that `ComponentOne` controls will not work in an environment where reflection is not allowed. Complete the steps in the [Modifying or Editing the Config File](#) topic to create or modify a policy file before completing the following.

ReflectionPermission

By default `ReflectionPermission` is not available in a medium trust environment. `ComponentOne` ASP.NET Wijmo controls require reflection permission because `LicenseManager.Validate()` causes a link demand for full trust.

To add reflection permission, complete the following:

1. Open the `web_mediumtrust.config` file or a file created based on the `web_mediumtrust.config` file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
  <SecurityClass Name="ReflectionPermission"
Description="System.Security.Permissions.ReflectionPermission,
mscorlib, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"/>
```

```
...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
    <IPermission
      class="ReflectionPermission"
      version="1"
      Flags="ReflectionEmit,MemberAccess" />
    ...
  </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the `web_mediumtrust.config` file.

OleDbPermission

By default `OleDbPermission` is not available in a medium trust environment. This means you cannot use the ADO.NET managed OLE DB data provider to access databases. If you wish to use the ADO.NET managed OLE DB data provider to access databases, you must modify the `web_mediumtrust.config` file.

To add `OleDbPermission`, complete the following steps:

1. Open the `web_mediumtrust.config` file or a file created based on the `web_mediumtrust.config` file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
  <SecurityClass Name="OleDbPermission"
Description="System.Data.OleDb.OleDbPermission, System.Data,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
  ...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
  <PermissionSet class="NamedPermissionSet" version="1"
Name="ASP.Net">
    <IPermission class="OleDbPermission" version="1"
Unrestricted="true"/>
    ...
  </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the `web_mediumtrust.config` file.

FileIOPermission

By default, FileIOPermission is not available in a medium trust environment. This means no file access is permitted outside of the application's virtual directory hierarchy. If you wish to allow additional file permissions, you must modify the `web_mediumtrust.config` file.

To modify FileIOPermission to allow read access to a specific directory outside of the application's virtual directory hierarchy, complete the following steps:

1. Open the `web_mediumtrust.config` file or a file created based on the `web_mediumtrust.config` file.
2. Add the following `<SecurityClass>` tag after the `<SecurityClasses>` tag so that it appears similar to the following:

```
<SecurityClasses>
    <SecurityClass Name="FileIOPermission"
        Description="System.Security.Permissions.FileIOPermission, mscorlib,
        Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
    ...
</SecurityClasses>
```

3. Add the following `<IPermission>` tag after the `<NamedPermissionSets>` tag so it appears similar to the following:

```
<NamedPermissionSets>
    <PermissionSet class="NamedPermissionSet" version="1"
        Name="ASP.Net">
        ...
        <IPermission class="FileIOPermission" version="1"
            Read="C:\SomeDir;$AppDir$" Write="$AppDir$" Append="$AppDir$"
            PathDiscovery="$AppDir$" />
        ...
    </PermissionSet>
</NamedPermissionSets>
```

4. Save and close the `web_mediumtrust.config` file.

End-User License Agreement

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

Licensing FAQs

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

What is Licensing?

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

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

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

How does Licensing Work?

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

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

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

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

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

These files are automatically added to the project.

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

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

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

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

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

Common Scenarios

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

Creating components at design time

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

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

Creating components at run time

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

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

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

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

Inheriting from licensed components

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

- Add a LicenseProvider attribute to the component.

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

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

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

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

Please note, that C1 licensing will not accept a run-time license for a derived control if the run-time license is embedded in the same assembly as the derived class definition, and the assembly is a DLL. This restriction is necessary to prevent a derived control class assembly from being used in other applications without a design-time license. If you create such an assembly, you will need to take one of the actions previously described create a component at run time.

Using licensed components in console applications

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

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

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

Using licensed components in Visual C++ applications

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

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

1. Build the C++ project as usual. This should create an .exe file and also a licenses.licx file with licensing information in it.
2. Copy the licenses.licx file from the app directory to the target folder (Debug or Release).
3. Copy the C1Lc.exe utility and the licensed dlls to the target folder. (Don't use the standard lc.exe, it has bugs.)
4. Use C1Lc.exe to compile the licenses.licx file. The command line should look like this:

```
c1lc /target:MyApp.exe /complist:licenses.licx /i:C1.Win.C1FlexGrid.dll
```
5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select properties, and go to the Linker/Command Line option. Enter the following:

```
/ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses
```
6. Rebuild the executable to include the licensing information in the application.

Using licensed components with automated testing products

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

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

For example:

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

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

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

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

Troubleshooting

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

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

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

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

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

If that fails follow these steps:

1. Open the affected project.
2. Select an instance of the updated component.
3. In the Visual Studio Properties window, change any property. It does not matter which property you change; you can change it back to the previous value.
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

Alternative 1: Follow these steps:

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

Alternative 2: Follow these steps:

1. Open the affected project.
2. Delete the licenses.licx file from the project.
3. Add a new instance of the updated component to the form.
4. Rebuild and run the solution. The nag screen should not appear.
5. Remove the newly added component from the form.

Try each of these options multiple times, if necessary. If that still does not help, are you creating any of the controls in code rather than design-time? If so, you must add an entry for the control in the licenses.licx file (see <http://helpcentral.componentone.com/PrintableView.aspx?ID=1886> for more information). Also if this is a website, as opposed to an ASP.NET web application, please try right-clicking the licenses.licx file and selecting "Build Runtime Licenses" from the context menu.

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

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

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

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

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

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

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

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

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

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

Option 2 – Continue to use the components you have.

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

Technical Support

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

Some methods for obtaining technical support include:

- **[Online Resources](#)**
ComponentOne provides customers with a comprehensive set of technical resources in the form of FAQs, samples and videos, Version Release History, searchable Knowledge base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**
This online support service provides you with direct access to our Technical Support staff via an [online incident submission form](#). When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.
- **Product Forums**
ComponentOne's [product forums](#) are available for users to share information, tips, and techniques regarding ComponentOne products. ComponentOne developers will be available on the forums to share insider tips and technique and answer users' questions. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.
- **Installation Issues**
Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the [online incident submission form](#) or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.
- **Documentation**
Microsoft integrated ComponentOne documentation can be installed with each of our products, and documentation is also available online. If you have suggestions on how we can improve our documentation, please email the [Documentation team](#). Please note that e-mail sent to the [Documentation team](#) is for

documentation feedback only. [Technical Support](#) and [Sales](#) issues should be sent directly to their respective departments.

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

Redistributable Files

ComponentOne Studio for ASP.NET Wijmo 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.Web.Wijmo.Controls.3.dll
- C1.Web.Wijmo.Controls.Design.3.dll
- C1.Web.Wijmo.Controls.4.dll
- C1.Web.Wijmo.Controls.Design.4.dll
- C1.Web.Wijmo.Extenders.3.dll
- C1.Web.Wijmo.Extenders.4.dll
- C1.C1Report.2.dll
- C1.C1Report.4.dll

Site licenses are available for groups of multiple developers. Please contact Sales@ComponentOne.com for details.

About This Documentation

Acknowledgements

Microsoft, Windows, Windows Vista, Visual Studio, and Microsoft Expression are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Firefox is a registered trademark of the Mozilla Foundation.

Safari is a registered trademark of Apple Inc.

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](#).

Namespaces

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

The general namespace for ComponentOne Web products is **C1.Web**. The following code fragment shows how to declare a **C1ReportViewer** using the fully qualified name for this class:

- Visual Basic

```
Dim ReportViewer As C1.Web.Wijmo.Controls.ReportViewer.C1ReportViewer
```

- C#

```
C1.Web.Wijmo.Controls.ReportViewer.C1ReportViewer ReportViewer;
```

Namespaces address a problem sometimes known as *namespace pollution*, in which the developer of a class library is hampered by the use of similar names in another library. These conflicts with existing components are sometimes called *name collisions*.

Fully qualified names are object references that are prefixed with the name of the namespace where the object is defined. You can use objects defined in other projects if you create a reference to the class (by choosing Add Reference from the Project menu) and then use the fully qualified name for the object in your code.

Fully qualified names prevent naming conflicts because the compiler can always determine which object is being used. However, the names themselves can get long and cumbersome. To get around this, you can use the Imports statement (**using** in C#) to define an alias — an abbreviated name you can use in place of a fully qualified name. For example, the following code snippet creates aliases for two fully qualified names, and uses these aliases to define two objects:

- Visual Basic

```
Imports C1ReportViewer = C1.Web.Wijmo.Controls.ReportViewer.C1ReportViewer  
Imports MyReportViewer = MyProject.Objects.ReportViewer.C1ReportViewer
```

```
Dim wm1 As C1ReportViewer
```

```
Dim wm2 As MyReportViewer
```

- C#

```
using C1ReportViewer = C1.Web.Wijmo.Controls.ReportViewer.C1ReportViewer;  
using MyReportViewer = MyProject.Objects.ReportViewer.C1ReportViewer;
```

```
C1ReportViewer wm1;
```

```
MyReportViewer wm2;
```

If you use the **Imports** statement without an alias, you can use all the names in that namespace without qualification provided they are unique to the project.

Creating an ASP.NET Project

ComponentOne Studio for ASP.NET Wijmo requires Visual Studio 2008 or later and .NET Framework 3.0 or later for your Web applications. **Studio for ASP.NET Wijmo** does not require AJAX extensions; however, you can install them if you want to use AJAX in your project. See the following table for more details on installing AJAX extensions.


Visual Studio 2010

You can build Ajax-enabled ASP.NET projects by downloading the AJAX Control Toolkit from [CodePlex](#) and dragging-and-dropping the controls from the Visual Studio Toolbox onto an ASP.NET Web Forms page.

Visual Studio 2008, .NET Framework 3.5 You can easily create an AJAX-enabled ASP.NET project without installing separate add-ins because the framework has a built-in AJAX library and controls.

Visual Studio 2008, .NET Framework 3.0 You must install the ASP.NET AJAX Extensions 1.0, which can be found at <http://www.asp.net/ajax/downloads/archive/>. Then you can create an AJAX 1.0-Enabled ASP.NET 2.0 Web site or application.

The following topics explain how to create projects in Visual Studio 2010 and 2008.

- **Creating a Web Site/Application Project in Visual Studio 2010** 

To create a new Web site/application project in Visual Studio 2010, complete the following steps.

1. If you are creating an AJAX project, download the AJAX Control Toolkit from [CodePlex](#) and extract the files.
2. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.
3. Select a .NET Framework in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.
4. Under **Project Types**, choose either **Visual Basic** or **Visual C#** and then select **Web**. Note that one of these options may be located under **Other Languages**.
5. Select a language, and in the list of templates, select **ASP.NET Web Site/ Application**.
6. Specify a location and then click **OK**.

Note: The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

A new Web project is created at the root of the Web server you specified.

7. If you are creating an AJAX project, right-click the Toolbox (create a new tab if you like), select **Choose Items** and browse to find the **AjaxControlToolkit.dll**. You can begin dragging toolkit controls to your page. Note that if you do not see the toolkit controls in the Toolbox, make sure you selected the .NET Framework that corresponds with the version of the toolkit you downloaded.

- **Creating a Web Site/Application Project in Visual Studio 2008** 

To create a Web site/application project in Visual Studio 2008, complete the following steps:

1. From the **File** menu, select **New | Web Site/Project**. The New Web Site/New Project dialog box opens.
2. Select .NET Framework 3.5 or 3.0 in the upper right corner. Note that if you choose .NET Framework 3.0, you must install the [extensions](#) first.
3. Select a language, and in the list of templates, select **ASP.NET Web Site/ Application** or **AJAX 1.0-Enabled ASP.NET 2.0 Web Site/ Application**.
4. Specify a location and then click **OK**.

Note: The Web server must have IIS version 6 or later and the .NET Framework installed on it. If you have IIS on your computer, you can specify http://localhost for the server.

A new Web project is created at the root of the Web server you specified.

Adding the ReportViewer for ASP.NET Wijmo Components to a Project

When you open Visual Studio, you will notice a **ComponentOne Studio for ASP.NET Wijmo Projects** tab containing the ComponentOne controls that have automatically been added to the Toolbox.

Note that you can manually add ComponentOne controls to the Toolbox at a later time.

Manually Adding the Studio for ASP.NET Wijmo controls to the Toolbox

When you install **ComponentOne Studio for ASP.NET Wijmo**, the following ReportViewer for ASP.NET Wijmo components will appear in the Visual Studio Toolbox customization dialog box:

- C1ReportViewer

To manually add **the Studio for ASP.NET Wijmo** controls to the Visual Studio Toolbox:

6. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary) and right-click it to open the context menu.
7. To make the Studio for ASP.NET Wijmo components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, Studio for ASP.NET Wijmo, for example.
8. Right-click the tab where the component is to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
9. In the dialog box, select the **.NET Framework Components** tab. Sort the list by Namespace (click the **Namespace** column header) and check the check boxes for all components belonging to namespace C1.Web.Wijmo.Controls.ReportViewer. Note that there may be more than one component for each namespace.
10. Click **OK** to close the dialog box. The controls are added to the Visual Studio Toolbox.

Adding Studio for ASP.NET Wijmo Controls to the Form

To add **Studio for ASP.NET Wijmo** controls to a form:

1. Add them to the Visual Studio Toolbox.
2. Double-click each control or drag it onto your form.

Adding a Reference to the Assembly

To add a reference to the C1.Web.Wijmo.Controls.3 or C1.Web.Wijmo.Controls.4 assembly:

1. Select the **Add Reference** option from the **Website** menu of your Web Site project or from the **Project** menu of your Web Application project.
2. Select the most recent version of the **ComponentOne Studio for ASP.NET Wijmo** assembly from the list on the **NET** tab or browse to find the C1.Web.Wijmo.Controls.3.dll or C1.Web.Wijmo.Controls.4.dll file and click **OK**.
3. Select the **Form1.vb** tab or go to **View|Code** to open the Code Editor. At the top of the file, add the following **Imports** directive (**using** in C#):

```
Imports C1.Web.Wijmo.Controls
```

Note: This makes the objects defined in the **C1.Web.Wijmo.Controls.3(4)** assembly visible to the project. See [Namespaces](#) for more information.

Key Features

ComponentOne ReportViewer for ASP.NET Wijmo includes several unique features, including the following:

- **Pixel-perfect Reports in Any Browser**

C1ReportViewer uses WebPaper to create pixel perfect reports that are truly in a league of their own. ComponentOne WebPaper technology allows you to render reports in a high fidelity graphical form.

- **Crystal Reports**

C1ReportViewer supports features found in Microsoft Access and Crystal Reports. With the click of a button, import Access report files (MDB) and Crystal report files (RPT) using the C1ReportDesigner.

- **Microsoft Access and SQL Reporting Services Reports**

C1ReportViewer has extensive support of Microsoft reporting technologies. Both Access and SQL Server Reporting Services Reports are supported in the viewer. The support for RDL allows you to use existing enterprise reports embedded in your Web app with WebPaper technology.

- **Zooming**

By default, the report pages are zoomed to fit the whole page. You can change the zoom to show the pages in actual size, to fit the page width in the preview window, to fit the content width in the preview window, or to a custom zoom mode which is determined by the value of the **Zoom** property.

- **Exporting**

Export your reports directly to various portable formats: Excel, PDF, HTML, text, and images. This allows you to conveniently send your reports via e-mail or share them across your enterprise.

- **Auto-generate Outlines**

The outline tree is created automatically based on the report groups and can be used to navigate through the report in the Adobe Acrobat viewer. Each node in the tree corresponds to a group header section in the report. Invisible group header sections do not generate outline entries in the PDF document.

- **Paged or Flowing Navigation**

The viewing pane of **C1ReportViewer** supports both paged navigation and scrolling navigation. Both views can be used easily with the click of a button. It gives your end-users the option to choose which they prefer.

- **Built-in Search**

The **C1ReportViewer** control comes with a built-in search pane that queries an auto-generated index of your report. This feature gives your end-users a powerful search tool in reports without writing a single line of code. It also highlights and builds a menu of the search results to help your end-users find what they are looking for.

- **Efficient Report Cache**

Experience extremely fast response times with **C1ReportViewer's** efficient report cache. Each time the **C1ReportViewer** control renders a report it is compressed and stored in the server, eliminating the need to re-query the database and regenerate the report when it is needed again. This reduces server loads and results in extremely fast response times at low memory cost.

- **Powerful Printing Capabilities**

The **C1ReportViewer** control has built-in print features including a print button that launches a custom print dialog box. The print dialog box allows your end-users to change how their report gets printed and even shows them a live preview of the document.

- **C1ReportDesigner Application**

Quickly create, edit, preview, load, and save report definition files without writing a single line of code. The familiar Microsoft Access-like user interface of the **C1ReportDesigner** application yields fast adaptation. The designer's new Microsoft Office 2007 Ribbon-style UI organizes related commands into groups enabling you to design reports faster than ever before.

- **Banded Report Model**

Reports uses a banded report model based on groups, sections, and fields. The banded report model allows for a highly organized report layout.

- **30+ Built-in Report Templates**

The enhanced report designer application now includes 34 report templates. Simply select a report from all of the **CIReportViewer** samples and you get a professionally styled report in minutes. No coding required - your colorful report is just a click away!

- **Flexible Data Binding**

Specify a connection string and an SQL statement in your report definition and **CIReportViewer** will load the data automatically for you. Optionally, use XML files, custom collections, and other data sources.

- **Nested Reports**

CIReportViewer may contain nested reports to arbitrary levels (subreports). You can use the main report to show detailed information and use subreports to show summary data at the beginning of each group.

ReportViewer for ASP.NET Wijmo Quick Start

The following quick start is designed to quickly familiarize you with the features for the C1ReportViewer control. In this quick start you'll create a simple Web site that uses the C1ReportViewer control to view a report. Note that in this example you'll use the **CommonTasks.xml** example file which should be installed in the ComponentOne Samples folder.

Step 1 of 3: Adding C1ReportViewer to the Page

The **C1ReportViewer** control is easy to create and setup. In this step you'll create a new application and add a **C1ReportViewer** control to the page so you can see just how easy it is.

Complete the following steps:

1. From the Visual Studio **File** menu select **New | Project**. The **New Project** dialog box will appear.
2. In the **New Project** dialog box expand a language in the left-hand pane and select **Web**. In the right pane, choose **ASP.NET Empty Web Application**, enter a **Name** for your application, and select **OK**. A new application will be created.
3. In the Solution Explorer, right-click the project and choose **Add Reference**.
4. In the Add Reference dialog box, locate and select the **C1.Web.Wijmo.Controls** and **C1.Web.Wijmo.Controls.Design** assemblies and click **OK**. The references will be added.
5. Right-click the project in the Solution Explorer and from the context menu choose **Add | New Item**.
6. In the **Add New Item** dialog box choose **Web Form** from the list of templates, name the item "Default.aspx", and click **Add**. The new page should open.
7. In the Solution Explorer, right click the project name and choose **Add | New Folder**. Name the new folder "App_Data".
8. Click the **Design** tab located below the Document window to switch to Design view.
9. Navigate to the Visual Studio Toolbox and double-click the **C1ReportViewer** control to add it to the page. The **C1ReportViewer** control is a fully-functioning report viewer.

Note that the **C1ReportViewer** control is a fully functioning report viewer. For detailed information about the parts of the control, see [C1ReportViewer Elements](#) (page 25).

Now that you've created your Web site and added the **C1ReportViewer** control to the page, the next step is to add a report to display in the control.

Step 2 of 3: Adding Reports to the Control

In the last step you created a new application and added the **C1ReportViewer** to the project. In this step you'll add a data source to your application and load a report into the **C1ReportViewer** control without using any code at all. Note that in this step you'll use the **CommonTasks.xml** example file which should be installed in the **ComponentOne Samples** folder.

Complete the following steps:

1. Navigate to the Solution Explorer window, right-click the project name, and choose **New Folder**.
2. Name the new folder "~/tempReports".

3. In the Solution Explorer, right-click the **App_Data** folder and select **Add Existing Item**. The **Add Existing Item** dialog box will appear.
4. In the **Add Existing Item** dialog box locate the **C1Nwind.mdb** file and click the **Add** button to close the dialog box and add the file to the project.

By default, the **C1NWind.mdb** file will be located in the **Documents** or **My Documents** folder at **ComponentOne Samples\Common**.

5. In the Solution Explorer, right-click the project name and select **Add Existing Item**. The **Add Existing Item** dialog box will appear.
6. In the **Add Existing Item** dialog box locate the **CommonTasks.xml** report definition file and click the **Add** button to close the dialog box and add the file to the project.

By default, the **CommonTasks.xml** file will be located in the **Documents** or **My Documents** folder at **ComponentOne Samples\C1Report\C1Report\XML\CommonTasks**.

Note: Report definition files are created with a separate utility, the **C1ReportDesigner**. The **C1ReportDesigner** works like the Access report generator, and is the same designer that ships with the **C1Report** component. The designer lets you create new reports from scratch or import existing ones from Microsoft Access and Crystal Reports. For more information on the **C1ReportDesigner**, see *Working with the C1ReportDesigner* in the **Reports for WinForms** documentation.

7. Click once on the **C1ReportViewer** control so that it is selected and navigate to the Properties window.
8. In the Properties window, set the **FileName** property to "**~/CommonTasks.xml**".
9. In the Properties window, set the **ReportName** property to "**01: Alternating Background (Greenbar report)**".

The **C1ReportViewer** will now display this report at run time.

In this step you added data source to your application and loaded a report into the **C1ReportViewer** control without using any code. In the next (and last) step you'll run your application and see the **C1ReportViewer** control in action.

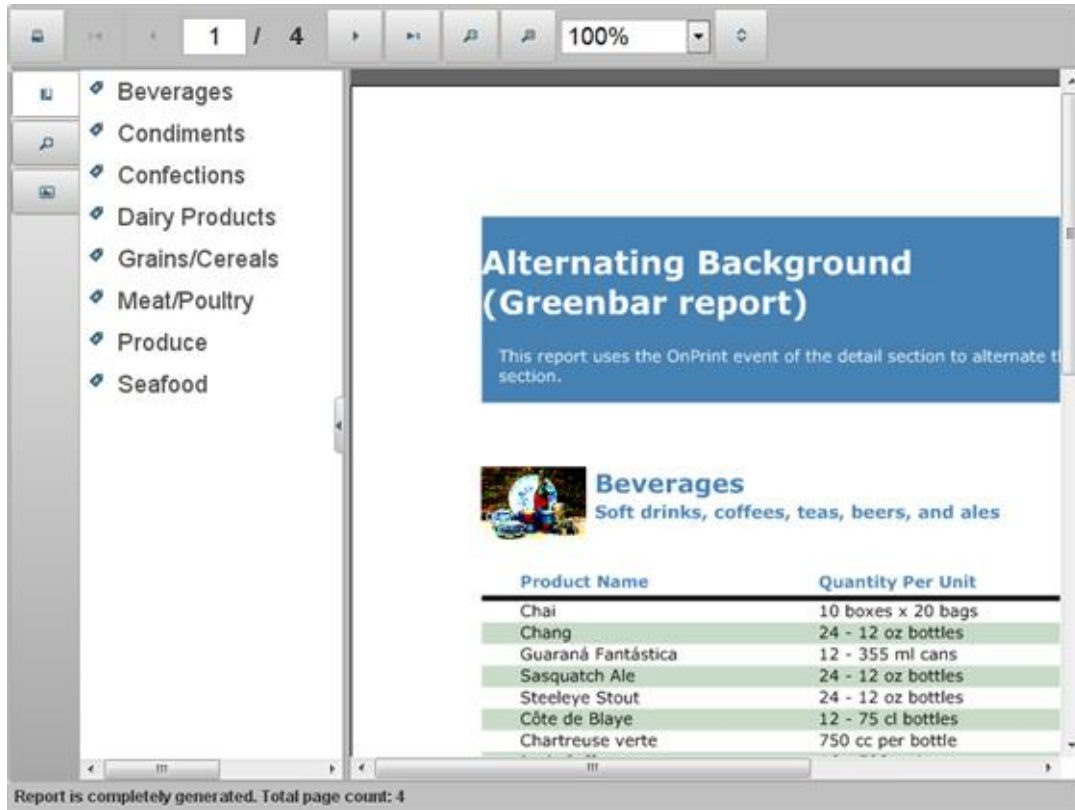
Step 3 of 3: Running the Application

In the previous steps you created and added a report to view in the **C1ReportViewer** control. In this last step you'll run your application and view the **C1ReportViewer** control at run time.

Complete the following steps:

1. Run your application.

The **C1ReportViewer** control displays the **01: Alternating Background (Greenbar report)** report in the **CommonTasks.XML** file.



2. Click the **Next** arrow button.
Observe that the displayed report moves to the next page.
3. Click an item in the contents, for example **Confections**.
Observe that the **C1ReportViewer** control now displays that section of the report.
4. Click the **Zoom Out** button a few times.
Observe that the displayed report appears smaller.
5. Click the **Continuous View** button:
Now if you scroll through the report, the report scrolls continuously.

Congratulations, you've completed the **Report Viewer for ASP.NET Wijmo** quick start! You've learned how to add a report definition file to view a report in the **C1ReportViewer** control. You also explored some of the run-time interactions possible with the control.

Wijmo Top Tips

The following tips may help you troubleshoot when working with **Studio for ASP.NET Wijmo**.

Tip 1: Prevent poor page rendering in quirks mode by editing the meta tag to fix rendering.

If a user's browser is rendering a page in quirks mode, widgets and controls may not appear correctly on the page. This is indicated by a broken page icon in the address bar. In **Compatibility View**, the browser uses an older rendering engine.



Users can set this view that causes the issue. To prevent rendering in quirks mode, you can force the page to render with the latest browser. Add the following meta tag to the header of the page:

```
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
```

Tip 2: Prevent errors when upgrading ReportViewer for ASP.NET applications.

You must remove the **reportService.ashx** file from your project before upgrading **C1ReportViewer** to a newer version. A new version of this file will be automatically created when the control is initialized for the first time.

The C1ReportViewer Control

ReportViewer for ASP.NET Wijmo includes the **C1ReportViewer** control which is a control that allows users to view and browse reports or documents generated by ComponentOne reporting products in a Web application. The viewer's server side code uses **C1Report** assembly (C1.C1Report.2(4).dll, the same DLL that is used by **ComponentOne Reports for WinForms**) to generate documents and reports, and to serve pages to the viewer.

The report viewer's class is **C1.Web.Wijmo.Controls.C1Report.C1ReportViewer**. The viewer shows document and reports using an innovative technology that ComponentOne calls **Web Paper**. **Web Paper** technology allows the reports to be displayed with very high fidelity while preserving the ability to select, copy, and search text.

Note: For information about using **C1Report** and **C1PrintDocument**, see the [Reports for WinForms documentation](#).

Displaying Reports and Documents

The **C1ReportViewer** control can show any report or document that can be generated by **C1Report**. This is specified via the following three public properties on the viewer:

- **FileName:** Gets or sets the report or document file name.
- **ReportName:** Gets or sets the name of the report.

The **FileName** and **ReportName** properties can be set at design time.

The following kinds of documents can be shown by the **C1ReportViewer** control:

- A **C1Report** loaded from an XML file (such as the reports in the CommonTasks.xml sample shipped with C1Report). To specify this type of document, the FileName and ReportName properties should be assigned. For example:
 - Visual Basic


```
C1ReportViewer1.FileName = "~/CommonTasks.xml"
C1ReportViewer1.ReportName = "Greenbar Report"
```
 - C#


```
C1ReportViewer1.FileName = "~/CommonTasks.xml";
C1ReportViewer1.ReportName = "Greenbar Report";
```
- A **C1PrintDocument** loaded from a C1D or C1DX file. To specify this, the FileName property should be set to the name of the .cld/.cldx file, for example:
 - Visual Basic


```
C1ReportViewer1.FileName = "~/MyDocument.cldx"
```
 - C#


```
C1ReportViewer1.FileName = "~/MyDocument.cldx";
```
- An RDL report loaded from an RDL report definition file. For example:
 - Visual Basic


```
C1ReportViewer1.FileName = "~/MyRdlReport.rdl"
```
 - C#


```
C1ReportViewer1.FileName = "~/MyRdlReport.rdl";
```

Note on C1Report/C1PrintDocument Licensing

Note how in the [Displaying Reports and Documents](#) (page 23) section, in-memory **C1Report** and **C1PrintDocument** objects were created with calls to method on the **C1ReportViewer** rather than using regular constructors:

- Visual Basic


```
Dim doc As C1PrintDocument = Me.C1ReportViewer1.CreateC1PrintDocument()
Dim rep As C1.C1Report.C1Report = Me.C1ReportViewer1.CreateC1Report()
```
- C#


```
C1PrintDocument doc = this.C1ReportViewer1.CreateC1PrintDocument();
C1.C1Report.C1Report rep = this.C1ReportViewer1.CreateC1Report();
```

Using the **CreateC1Report** and **CreateC1PrintDocument** methods ensures that, as long as your **C1ReportViewer** control is licensed, the instances of C1.C1Report.C1Report and C1.C1Preview.C1PrintDocument classes that you create in this way are also licensed, and you or your users will not see the nag screen about using an unlicensed copy of C1Report.

If you simply wrote:

- Visual Basic


```
Dim doc As New C1PrintDocument()
Dim rep As New C1.C1Report.C1Report()
```
- C#

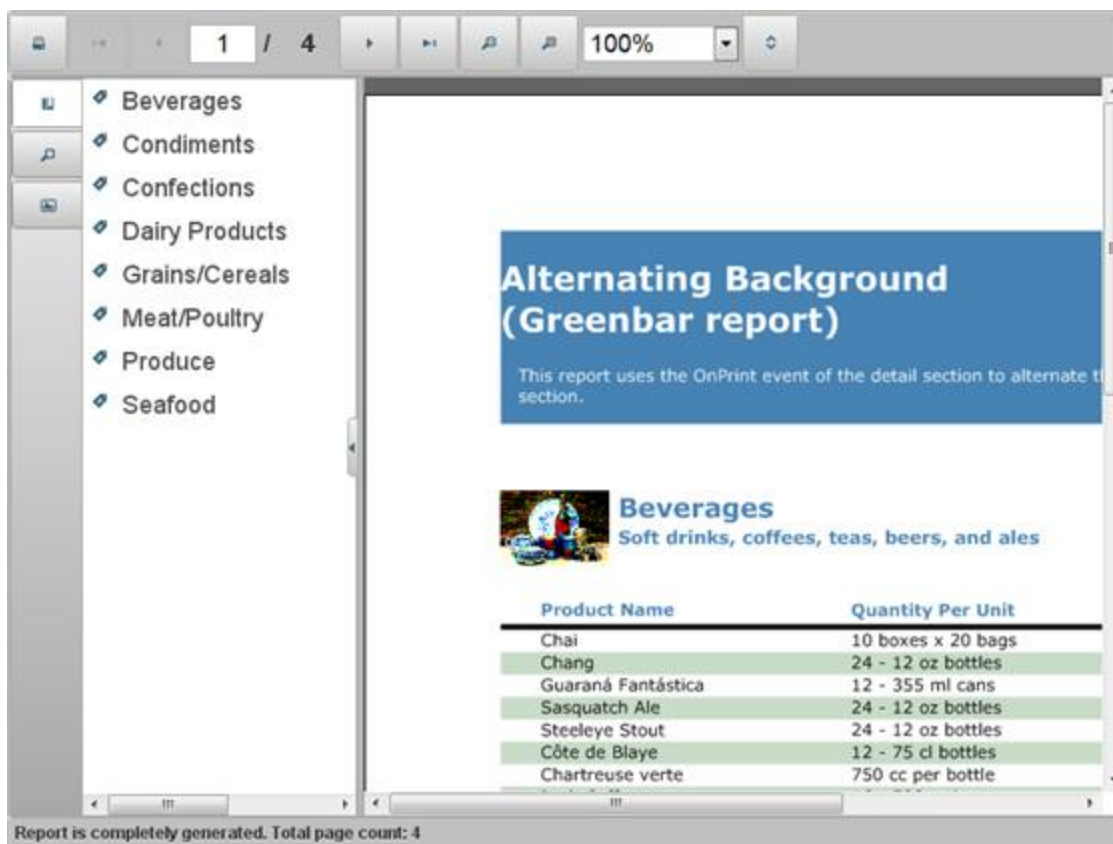

```
C1PrintDocument doc = new C1PrintDocument();
C1.C1Report.C1Report rep = new C1.C1Report.C1Report();
```

chances are that at run time, those calls would have generated nag screens because run-time licenses for C1.C1Report.C1Report and C1.C1Preview.C1PrintDocument were not embedded into the application.

So, whenever you need to create an instance of a `C1.C1Report.C1Report` or a `C1.C1Preview.C1PrintDocument` in code that you want to assign to the `C1DocumentViewer.Document`, use the appropriate `CreateC1Report` or `CreateC1PrintDocument` method to make sure the created instance is licensed properly.

C1ReportViewer Elements

This section provides a visual and descriptive overview of the elements that comprise the `C1ReportViewer` control. The `C1ReportViewer` control consists of several distinct elements. The `C1ReportViewer` control includes a toolbar, outline and search panes, and a report preview area. At run time the control appears similar to the following image:



This format should be familiar to end-users as it resembles how PDFs appear in viewers such as Adobe Acrobat Reader. The following sections describe each of these sections.

C1ReportViewer Toolbar

By default, a toolbar appears at the top of the `C1ReportViewer` control at run time. The toolbar allows users to print, save, navigate through, and change the display of the report at run time. If you do not want the toolbar to appear you can set the `ToolBarVisible` property to `False`.

The toolbar appears similar to the following image:



The toolbar includes the following options:

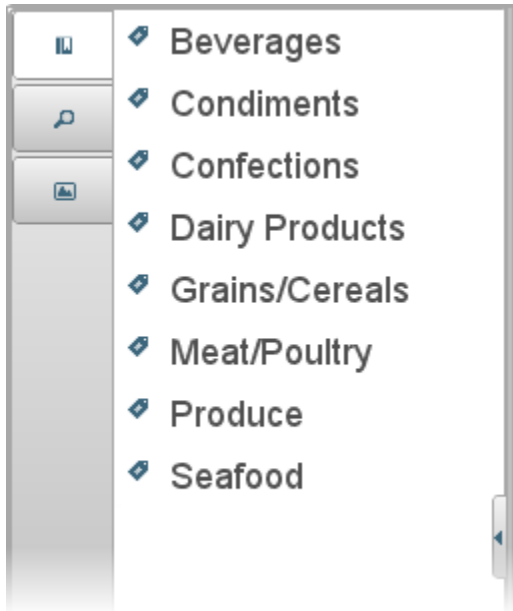
- **Print document:** The **Print document** button opens the Print dialog box where users can print a report. For more information, see [Printing a Report](#) (page 31).
- **First page:** Navigates to the first page of the report. See [Navigating a Report](#) (page 34) for more information.
- **Previous page:** Navigates to the previous page of the report. See [Navigating a Report](#) (page 34) for more information.
- **Page:** Indicates the currently selected page and allows users to select a page to view. See [Navigating a Report](#) (page 34) for more information.
- **Next page:** Navigates to the next page of the report. See [Navigating a Report](#) (page 34) for more information.
- **Last page:** Navigates to the last page of the report. See [Navigating a Report](#) (page 34) for more information.
- **Zoom Out:** The **Zoom Out** button decreases the size of the viewed report. See [Zooming a Report](#) (page 33) for more information.
- **Zoom In:** The **Zoom In** button increases the size of the viewed report. See [Zooming a Report](#) (page 33) for more information.
- **Current Zoom:** Indicates the current zoom level and allows the user to choose from predefined zoom options. See [Zooming a Report](#) (page 33) for more information.
- **Continuous View:** Lets the user view a report as a continuous series of pages. For more information, see [Changing Report Flow](#) (page 33).

For more information about using the toolbar at run time, see [Run-Time Interaction](#) (page 31).

C1ReportViewer Navigation Panes

By default, three tools panes appear on the left side of the **C1ReportViewer** control: an **Outline** pane, a **Search** pane, and a **Thumbs** pane. The Outline pane functions as a table of contents and lists sections of the displayed report so users can skip to a particular place in the report. The Search pane allows users to search the displayed report for particular words or phrases. The Thumbs pane displays thumbnail images of each page in the report so users can visually navigate to specific images or sections in the document.

The Outline pane of a report appears similar to the following image:



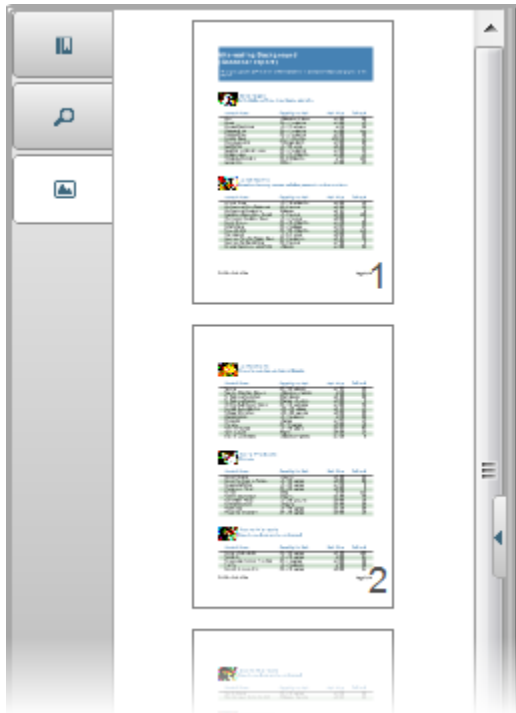
By clicking one of these options, users can skip to that section of the report. See [Navigating a Report](#) (page 34) for more information.

The Search pane appears similar to the following image:



Users can search for a word or phrase at run time and then jump to instances of that word or phrase in the report. For more information, see [Searching a Report](#) (page 35).

The Thumbs pane appears similar to the following image:



C1ReportViewer Preview Pane

By default, a preview pane appears in the right side of the **C1ReportViewer** control. The preview pane allows users to view the report that that is currently displayed in the control. For example, the preview pane appears similar to the following image with a report zoomed out and displayed:



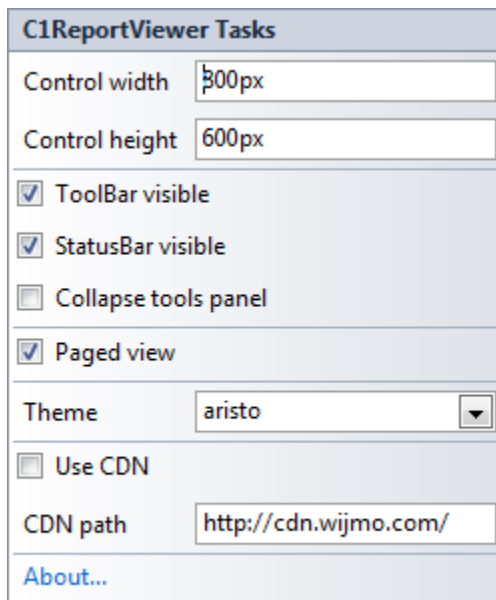
Design-Time Support

The following sections describe how to use **C1ReportViewer**'s design-time environment to configure the C1ReportViewer control.

C1ReportViewer Smart Tag

In Visual Studio, the **C1ReportViewer** control includes a smart tag. A smart tag represents a short-cut tasks menu that provides the most commonly used properties in **C1ReportViewer**. The **C1ReportViewer** control provides quick and easy access to common properties through its smart tag.

To access the **C1ReportViewer Tasks** menu, click on the smart tag in the upper-right corner of the **C1ReportViewer** control. This will open the **C1ReportViewer Tasks** menu, which appears like the following image:



The **C1ReportViewer Tasks** menu operates as follows:

- **Control width**
Determines the control's width at run time. The default width is **800px**.
- **Control height**
Determines the control's height at run time. The default height is **600px**.
- **ToolBarVisible**
When the **ToolBarVisible** check box is checked (default) the **ToolBarVisible** property is set to **True** and the **ReportViewer**'s toolbar is visible on the **C1ReportViewer** control at run time. Uncheck the **ToolBarVisible** check box if you do not want the toolbar displayed.
- **StatusBarVisible**
When the **StatusBarVisible** check box is checked (default) the **StatusBarVisible** property is set to **True** and the **ReportViewer**'s status bar is visible on the **C1ReportViewer** control at run time. Uncheck the **StatusBarVisible** check box if you do not want the status bar displayed.

- **Collapse tools panel**

When the **Collapse tools panel** check box is checked the CollapseToolsPanel property is set to **True** and the tools pane, which includes the outline, search, and thumbs panes, is appears collapsed on the C1ReportViewer control.

- **Paged view**

When the **Paged view** check box is un-checked the PagedView property is set to **False** and users can scroll from page to page. When the PagedView property is set to **True** (default) users must page through the document instead.

- **Theme**

Clicking the **Theme** drop-down box allows you to select from various visual schemes. For more information about available visual styles, see [Themes](#) (page 36).

- **Use CDN**

Selecting the **Use CDN** check box will indicate that the widget extender must load client resources from a content delivery network. By default this box is not checked.

- **CDN Path**

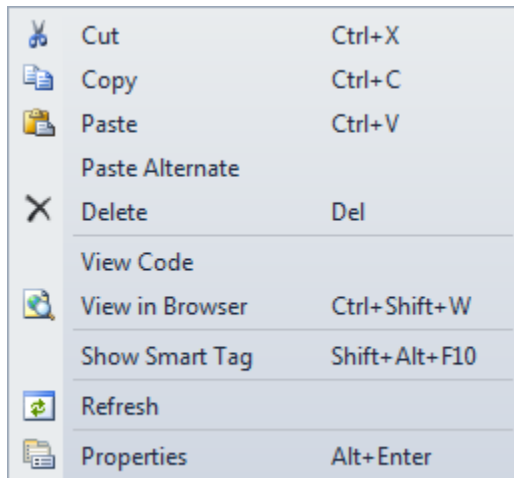
Indicates the path for the content delivery network. Enter a URL here to change the path.

- **About**

Clicking on the **About** item displays the **About** dialog box, which is helpful in finding the version number of **ReportViewer for ASP.NET Wijmo** and online resources.

C1ReportViewer Context Menu

Right-click anywhere on the list to display the C1ReportViewer context menu, which is a context menu that Visual Studio provides for all .NET controls. It will appear similar to the following image:



The context menu commands operate as follows:

- **Show Smart Tag**

Clicking this item shows the **C1ReportViewer Tasks** menu. For more information on how to use the smart tag and available features in the Tasks menu, see [C1ReportViewer Smart Tag](#) (page 29).

Run-Time Interaction

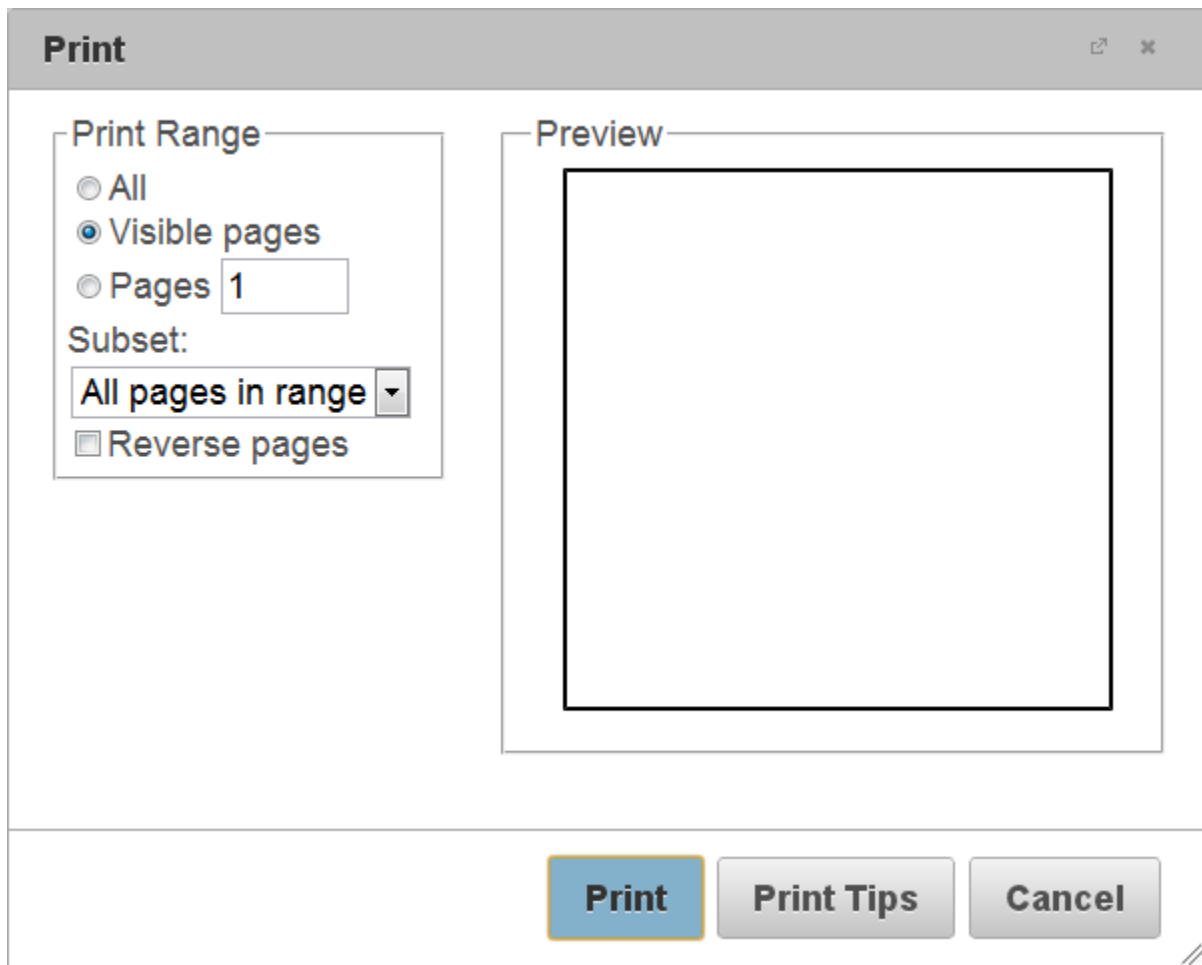
The following topics detail how to interact with the **CIReportViewer** control at run time. You'll learn how users can navigate and interact with displayed reports.

Printing a Report

At run time, users can easily print a report by clicking the **Print report** button in the toolbar:



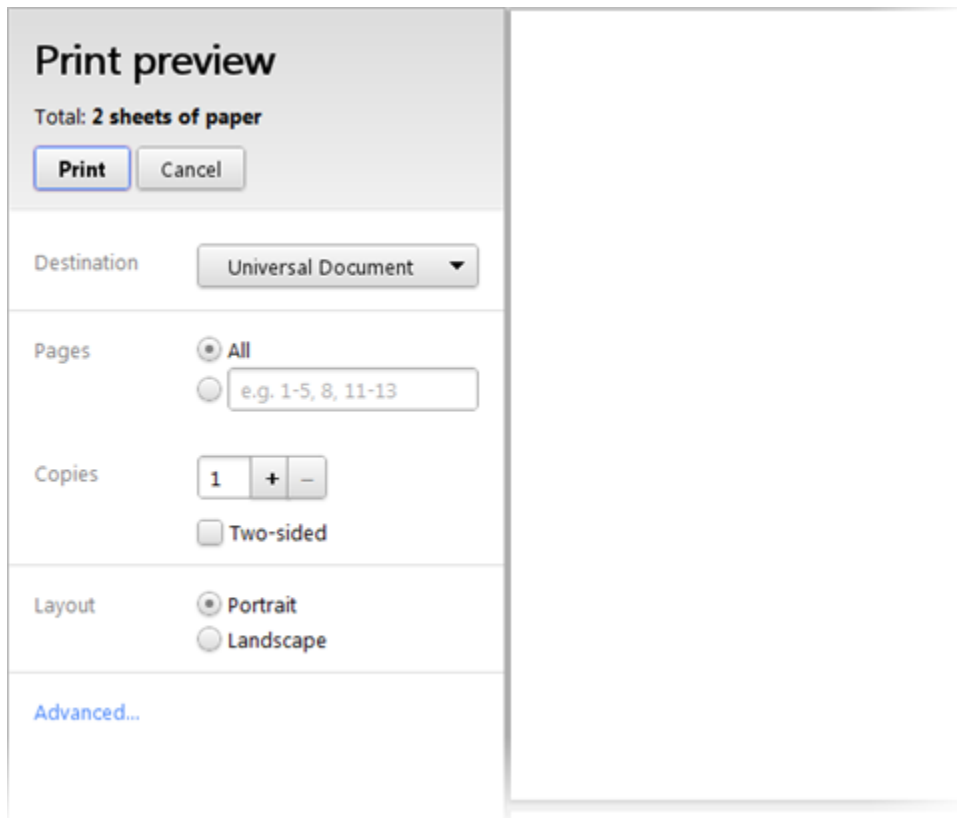
The **Print** dialog box will then appear:



The **Print** dialog box includes the following options:

- **Print Range:** Choose whether you want **All** pages displayed, only **Visible pages** (default), or only certain **Pages**. The **Subset** drop-down box lets you choose to print **All pages in range** (default), **Odd pages only**, or **Even pages only**. The **Reverse pages** check box lets you choose to reverse the order pages are displayed if selected.
- **Print:** The **Print** button brings up the **Print preview** screen from which you can print the report.
- **Print Tips:** The **Print Tips** button displays tips for printing reports to help users print documents at run time.
- **Cancel:** The **Cancel** button closes the **Print** dialog box without printing the report.

Once you select the **Print** button, the **Print preview** screen will appear:



The **Print** dialog box includes the following options:

- **Print:** The **Print** button brings up the **Print preview** screen from which you can print the report.
- **Cancel:** The **Cancel** button closes the **Print** dialog box without printing the report.
- **Destination:** The printer the report will print to.
- **Pages:** The pages to print, select **All** or choose an individual page or page range.
- **Copies:** The number of copies to print and if pages should be **two-sided**. By default one copy is printed and pages are only printed on one side.
- **Layout:** If pages should be printed in **Portrait** or **Landscape** mode.
- **Advanced:** Opens the **Print** dialog box to configure advanced printer options.

- **Preview Pane:** The Preview pane to the right of the menu lets you preview the pages that will be printed.

Printing Tips

The following tips will help users when printing documents:

- Make sure that you have configured the default printer correctly.
- Make sure that preview area contains the correct content; if needed wait until preview area has updated successfully.
- Be aware that content inside the preview area of **C1ReportViewer** will be printed 'as is'.

Note on printing: while the Print button provides a convenient and quick way to print all or part of a document, it should be understood that it has significant limitations. Anything printed from within a Web browser is subject to the browser's formatting, page headers and footers, and so on. PDF documents created by **C1ReportViewer** should be identical in appearance to the documents in the viewer – but should print much better as they may be printed avoiding limitations imposed by the Web browser.

Changing Report Flow

At run time, users can easily change the flow of a report by clicking the **Continuous view** button on the toolbar:



The **Continuous view** button changes the way a report is displayed. When selected, the PagedView property is set to **False** and users can scroll from page to page. When the PagedView property is set to **True** (default) users must page through the document instead.

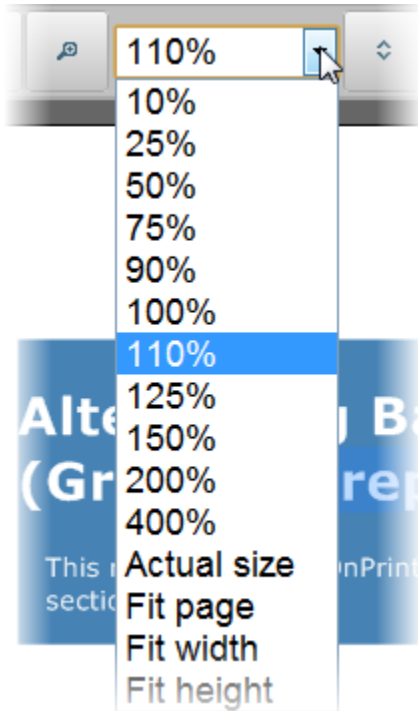
Zooming a Report

At run time, users can easily zoom in and out of a report by clicking the **Zoom In**, **Zoom Out**, and **Current Zoom** options on the toolbar:



The **Zoom** drop-down options on the toolbar include the following:

- **Zoom In:** Zooms the document in by 10%. If the document appears at 100% zoom, clicking this button will display the document at 110% zoom.
- **Zoom Out:** Zooms the document out by 10%. If the document appears at 100% zoom, clicking this button will display the document at 90% zoom.
- **Current Zoom:** Users can click the drop-down arrow and choose from one of the predefined zoom options:



Navigating a Report

At run time, users can navigate through the report document using several navigation options in the **CIReportViewer** control. Navigation options include an **Outline** pane and page navigation buttons on the toolbar.

The navigation buttons in the toolbar allow users to move from page to page in the report document and to skip to any specific page in the document:

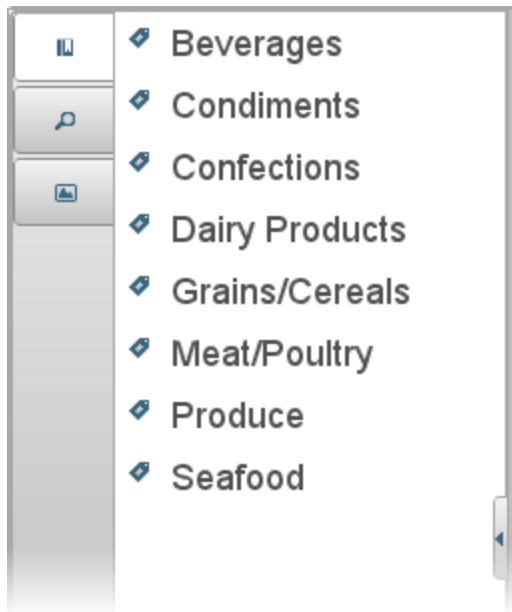


The **Navigation** options on the toolbar include the following:

- **First page:** Navigates to the first page in the report document. The option is available when any page other than the first page is displayed.
- **Previous page:** Navigates to the previous page in the report document. The option is available when any page other than the first page is displayed.
- **Page:** Displays the current page and the number of total pages. Users can enter a page number in the text box to navigate to that page.
- **Next page:** Navigates to the next page in the report document. The option is available when any page other than the last page is displayed.
- **Last page:** Navigates to the last page in the report document. The option is available when any page other than the last page is displayed.

At run time, users can also easily navigate sections of a report by clicking the **Outline** button in the tools pane.

The **Outline** pane will appear:



The **Outline** pane lists the sections of a report a user can easily navigate to. Users can click an item in the **Outline** pane to navigate to the part of the report document.

Searching a Report

At run time, users can easily search a report by clicking the **Search** button in the tools pane. This will open the **Search** panel.



The **Search** panel includes the following:

- **Text box:** Users can enter a word or phrase in the **Search** text box to search the report document for that word or phrase.
- **Case sensitive:** The **Case sensitive** check box determines if the text entered should be searched as case sensitive. For example, if this box is checked, searching for "Product" and "product" will produce different results.
- **Search:** Users can click the **Search** button to search for the text they entered in the text box.

- **Results:** If a search term is found in the document, it will be displayed in the Results window. The results window will list the number of matches found and the pages (with links to those pages) where the term was found.

ReportViewer for ASP.NET Wijmo Appearance

There are several options for customizing the appearance and layout of the C1ReportViewer control. The following sections describe how to change the appearance of the control through built-in themes as well as how to customize other elements of the C1ReportViewer control.

Themes

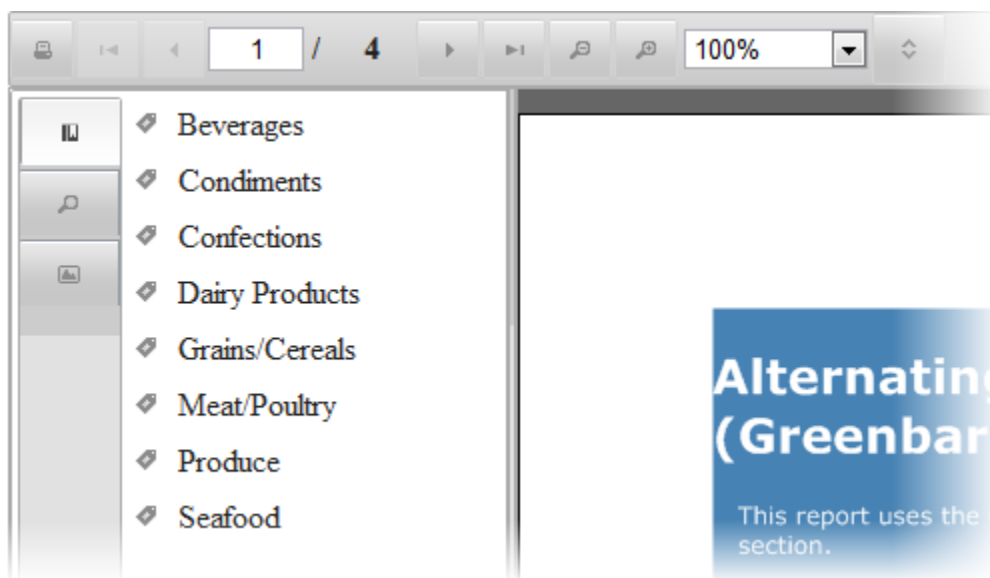
C1ReportViewer includes themes allowing you to easily change the control's appearance. The control includes several built-in themes allowing you to customize the control's appearance to your application. You can easily change themes from the **C1Dialog Tasks** menu, from the Properties window, and in code.

C1ReportViewer includes the following built-in themes:

- arctic
- aristo
- cobalt
- midnight
- rocket
- sterling

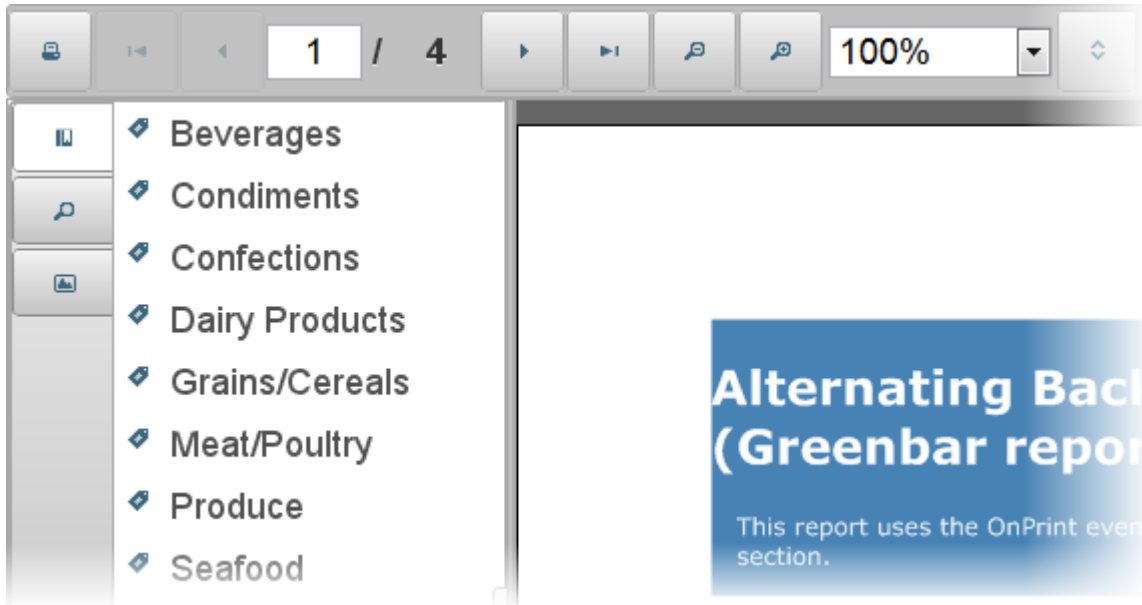
arctic

The following image displays the **arctic** theme:



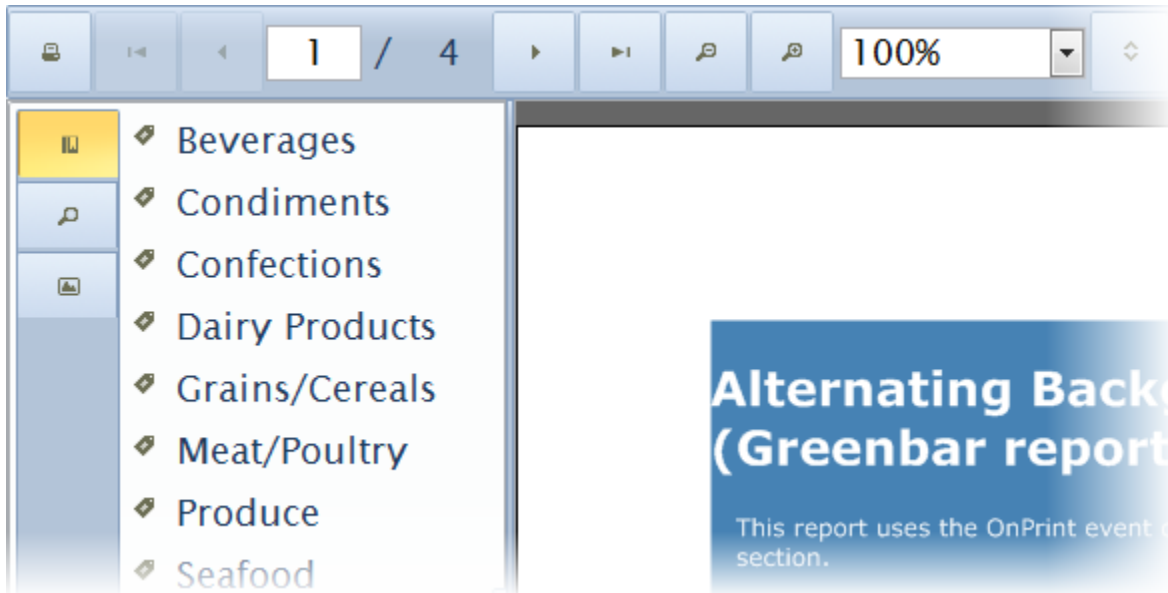
aristo

The following image displays the **aristo** theme. This is the default appearance of C1ReportViewer.



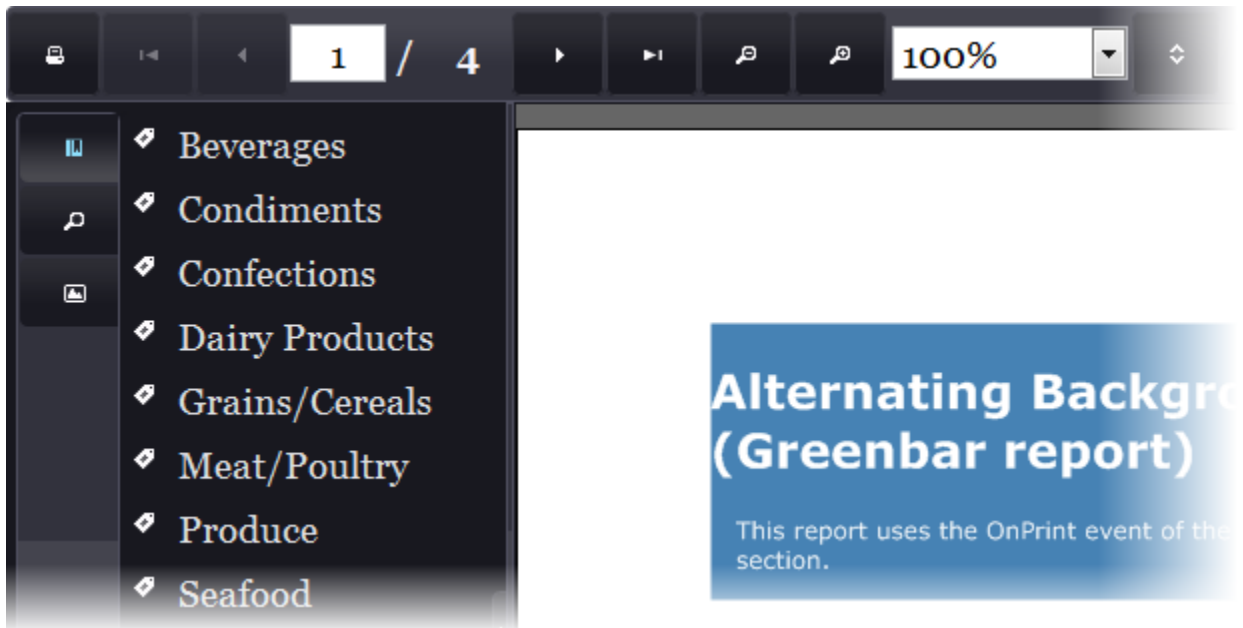
cobalt

The following image displays the **cobalt** theme:



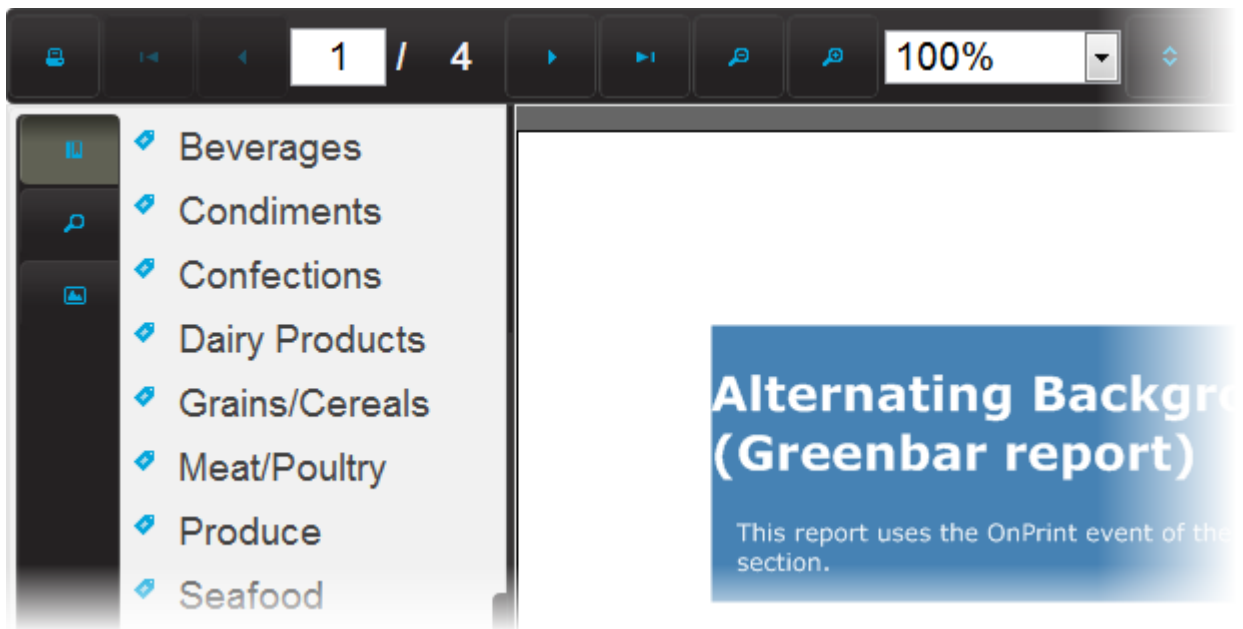
midnight

The following image displays the **midnight** theme:



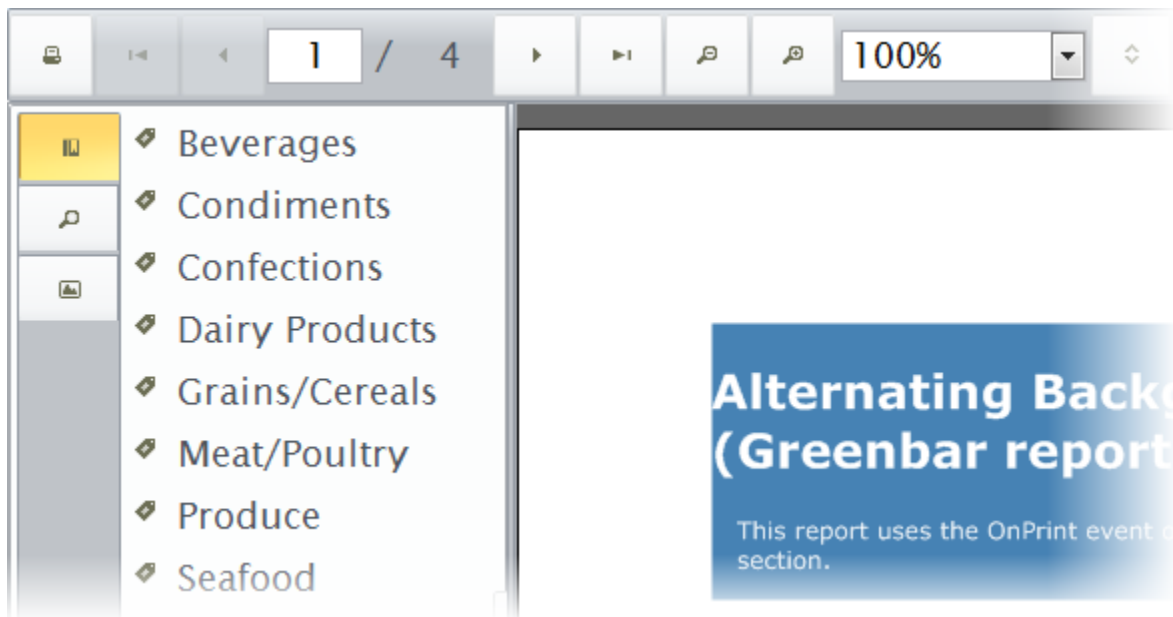
rocket

The following image displays the **rocket** theme:



sterling

The following image displays the **sterling** theme:



Changing Theme

You can change the theme of a C1ReportViewer at design time using the Properties window:

1. Click the C1ReportViewer control once to select it, and navigate to the Properties window.
2. In the Properties window, click the **Theme** drop-down arrow and select a style, for example **rocket**.
The theme you selected is applied to your grid.