
ComponentOne

Flash for .NET

Copyright © 1987-2010 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 Flash for .NET Overview	1
What's New in Flash for .NET	1
Installing Flash for .NET	1
Flash for .NET Setup Files	1
System Requirements	2
Installing Demonstration Versions	2
Uninstalling Flash for .NET	3
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?	6
Common Scenarios	7
Troubleshooting	9
Technical Support	10
Redistributable Files	11
About This Documentation	11
Namespaces	12
Creating a .NET Project	13
Adding the Flash for .NET Components to a Project	14
Migrating a Flash for .NET Project to Visual Studio 2005.....	15
Key Features	23
Create Canvas Documents with C1FlashCanvas	23
Creating Canvas Documents	24
Adding Text to C1FlashCanvas.....	24
Adding Images to C1FlashCanvas	26
Adding Graphics to C1FlashCanvas	27
Using Metafiles to Render Graphics	34
Create Movie Documents with C1FlashMovie	35
Understanding Frames and Graphical Objects	35
Creating Movie Documents.....	36
Create Slide Documents with C1FlashSlide	38
Creating Slide Documents	38
Using the C1FlashSlide Designer.....	38
Setting Common Slide Attributes.....	41
Setting the Header & Footer	41
Setting the Button Style	42
Flash for .NET Samples	43
Flash for .NET Task-Based Help	44
C1FlashCanvas Tasks.....	44
Drawing Text in C1FlashCanvas.....	44
Rendering Images Using C1FlashCanvas	48
Drawing Shapes or Filling Shapes in C1FlashCanvas	52
Transforming a Drawing Using C1FlashCanvas	54
C1FlashMovie Tasks	58
Creating Movie Documents that Rotate.....	59
C1FlashSlide Tasks	63

Creating Slide Documents with Navigation Buttons	63
C1WebFlash Class Overview	70
Getting Started with C1WebFlash	71
Creating an ASP.NET 2.0 Project.....	71
Adding the C1WebFlash Component to a Project.....	72
Creating a New Web Form.....	74
Setting the Start Page for Your Web Application	75
Adding the C1Flash Components to Your Web Application.....	76
Binding a Flash for .NET Component to the C1WebFlash Control.....	77
C1WebFlash Samples	78

ComponentOne Flash for .NET

Overview

Giving you the power to present your mission critical information in the form of vector graphics and animation, **ComponentOne Flash for .NET** allows you to create Adobe Flash (SWF) documents from your applications.

There are three WinForms components in the product package. Each is designed for different usage:

- **C1FlashCanvas** – a component similar to the .NET **Graphics** class. It provides methods for drawing content to a single frame, or canvas, of Flash. The coordinate in **C1FlashCanvas** is the logical pixel. If you want to generate Flash animations, you need to use **C1FlashMovie**.
- **C1FlashMovie** – a component that can be used to create multi-frames animation. You can add/remove/transform graphical objects to the frames. The coordinate in **C1FlashMovie** is a *twip* which is the measurement used by the Adobe SWF specification. In the SWF format, a *twip* is 1/20th of a logical pixel. A logical pixel is the same as a screen pixel when the movie is played at 100% - that is, without scaling.
- **C1FlashSlide** – a component that can be used to create a slide show in the Adobe Flash file format. Each page of the slide is an **FPage** class that provides the methods similar to those in the .NET Graphics class. With the powerful [Slide Designer](#) (page 38) you can layout and specify the properties of the UI elements in a convenient way, such as navigation buttons, page header, page footer, and page number.

Flash for .NET provides most of the graphical drawing abilities that SWF format supports, and a very important feature of **Flash for .NET** is its ease of use. Draw content to a single-frame Flash document, create Flash animation through a series of frames, or organize your Flash frames in a slide show.

What's New in Flash for .NET

This documentation was last revised for 2010 v1 on January 18, 2010.



Tip: A version history containing a list of new features, improvements, fixes, and changes for each product is available in HelpCentral at <http://helpcentral.componentone.com/VersionHistory.aspx>.

No new features have been added to **ComponentOne Flash for .NET**.

Installing Flash for .NET

The following sections provide helpful information on installing **ComponentOne Flash for .NET**.

Flash for .NET Setup Files

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

bin	Contains copies of all ComponentOne binaries (DLLs, EXEs).
Common	Contains support and data files that are used by many of the demo programs.
H2Help	Contains documentation for all Studio components.
C1Flash	Contains samples for ComponentOne Flash for .NET .

Samples

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

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

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

System Requirements

System requirements include the following:

Operating Systems:	Windows 2000 Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista™ Windows 7
Web Server:	Microsoft Internet Information Services (IIS) 6.0
Environments:	NET Framework 2.0 or later C# .NET Visual Basic .NET ASP.NET Internet Explorer 6.x or later Firefox 2.x or later Safari 2.x or later
Disc Drive:	CD or DVD-ROM drive if installing from CD

Installing Demonstration Versions

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

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

Uninstalling Flash for .NET

To uninstall **ComponentOne Flash for .NET**:

1. Open the **Control Panel** and select **Add or Remove Programs (Programs and Features in Vista)**.
2. Select **ComponentOne Studio for .NET 2.0** and click the **Remove** button.
3. Click **Yes** to remove the program.

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 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](#) (page 4).

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.
Give the new a name that indicates that it is your variation of medium trust; for example, AllowReflection_Web_MediumTrust.config.
3. Add the permissions that you want to grant. For examples, see [Adding Permissions](#) (page 4).
4. 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](#) (page 3) 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](#) (page 3) 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` 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 WinForms 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.)

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 select **Show All Files** from 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 Property 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 dialogs. This is the expected behavior since the test application typically does not contain the necessary licensing information, and there is no easy way to add it.

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

For example:

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

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

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

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

Troubleshooting

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

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

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

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

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

If that fails follow these steps:

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

Alternatively, follow these steps:

1. Open the project and go to the Solution Explorer window.

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

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

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

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

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

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

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

If your ASP.NET application uses WinForms user controls with constituent licensed controls, the 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/Support>.

Some methods for obtaining technical support include:

- **Online Support via [HelpCentral](#)**
ComponentOne HelpCentral provides customers with a comprehensive set of technical resources in the form of [FAQs](#), [samples](#), [Version Release History](#), [Articles](#), searchable [Knowledge Base](#), searchable [Online Help](#), and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**
This online support service provides you with direct access to our Technical Support staff via an online incident submission form. When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.
- **Peer-to-Peer Product Forums and Newsgroups**
ComponentOne peer-to-peer product [forums and newsgroups](#) are available to exchange information, tips, and techniques regarding ComponentOne products. ComponentOne sponsors these areas as a forum for users to share information. While ComponentOne does not provide direct support in the forums and newsgroups, we periodically monitor them to ensure accuracy of information and provide comments when appropriate. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.
- **Installation Issues**
Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the online incident submission form or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.
- **Documentation**
ComponentOne documentation is available with each of our products in HTML Help, Microsoft Help 2.0, and NetHelp format. The NetHelp version of the documentation is also available on [HelpCentral](#). 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 Flash for .NET 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.C1Flash.2.dll
- C1.Web.C1Flash.2.dll

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

About This Documentation

Acknowledgements

Microsoft, Visual Studio, Visual Basic, Windows XP, Windows 7, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

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](#) and Sandcastle® for .NET.

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 namespace for the **ComponentOne Flash for .NET** components is **C1.C1Flash**. The following code fragment shows how to declare a **Flash for .NET** component using the fully qualified name for this class:

- Visual Basic
`Dim Flash As C1.C1Flash.C1FlashCanvas`
- C#
`C1.C1Flash.C1FlashCanvas Flash;`

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*.

For example, if you create a new class named **C1Flash**, you can use it inside your project without qualification. However, the **C1Flash** assembly also implements a class called **C1Flash**. So, if you want to use the **C1Flash** class in the same project, you must use a fully qualified reference to make the reference unique. If the reference is not unique, Visual Studio .NET produces an error stating that the name is ambiguous. The following code snippet demonstrates how to declare these objects:

- Visual Basic
`' Define a new object based on your C1Flash class`
`Dim MyC1Flash As C1Flash`

`' Define a new C1Flash.C1FlashCanvas object`
`Dim Flash As C1.C1Flash.C1FlashCanvas`
- C#
`// Define a new object based on your C1Flash class`
`MyC1Flash As C1Flash;`

`// Define a new C1Flash.C1FlashCanvas object`
`Flash As C1.C1Flash.C1FlashCanvas;`

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 C1Flash = C1.C1Flash.C1FlashCanvas
Imports MyC1Flash = MyProject.Objects.C1Flash

Dim c1 As C1Flash
Dim c2 As MyC1Flash
```
- C#

```
using C1Flash = C1.C1Flash.C1FlashCanvas;
using MyC1Flash = MyProject.Objects.C1Flash;

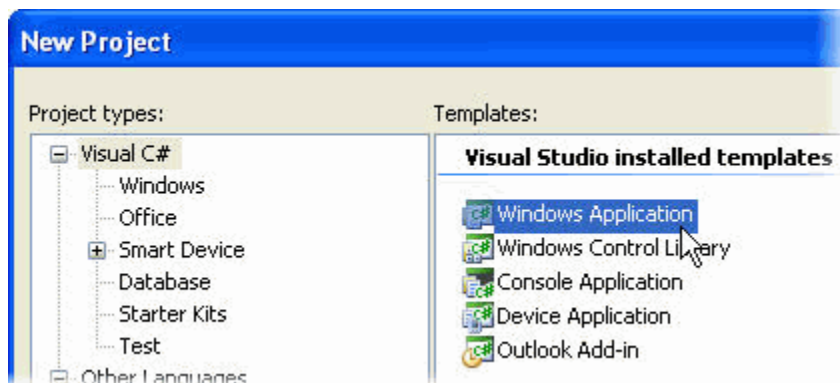
C1Flash c1;
MyC1Flash c2;
```

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 a .NET Project

To create a new .NET project, complete the following steps:

1. From the **File** menu in Microsoft Visual Studio .NET, select **New**, and click **Project**. The **New Project** dialog box opens.
2. Under **Project types**, choose either **Visual Basic** or **Visual C#**, and select **Windows Application** from the list of **Templates** in the right pane.



3. Enter or browse for a location for your application in the **Location** field and click **OK**.
A new Microsoft Visual Studio .NET project is created in the specified location. In addition, a new Form1 is displayed in the Designer view.
4. Double-click the desired **Flash for .NET** component from the Toolbox to add it to Form1. For information on adding a component to the Toolbox, see [Adding the Flash for .NET Components to a Project](#) (page 14).

Adding the Flash for .NET Components to a Project

When you install ComponentOne Studio for .NET 2.0, the **Create a ComponentOne Visual Studio 2008/2005 Toolbox Tab** checkbox is checked, by default, in the installation wizard. When you open Visual Studio, you will notice a **ComponentOne Studio for .NET 2.0** tab containing the ComponentOne controls has automatically been added to the Toolbox.

If you decide to uncheck the **Create a ComponentOne Visual Studio 2008/2005 Toolbox Tab** checkbox during installation, you can manually add ComponentOne controls to the Toolbox at a later time.

ComponentOne Flash for .NET provides the following controls:

- C1FlashCanvas
- C1FlashMovie
- C1FlashSlide

To use **Flash for .NET**, add these controls to the form or add a reference to the C1.C1Flash.2 assembly in your project.

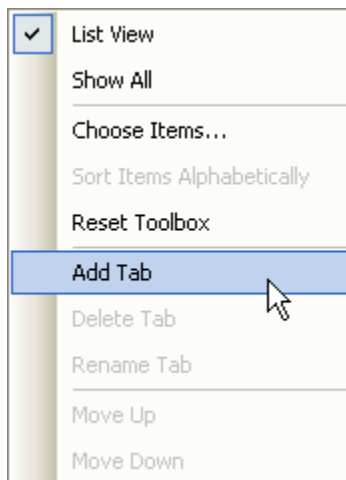
Manually Adding C1Flash to the Toolbox

When you install **C1Flash**, the following **C1Flash** components will appear in the Visual Studio Toolbox customization dialog box:

- C1FlashCanvas
- C1FlashMovie
- C1FlashSlide

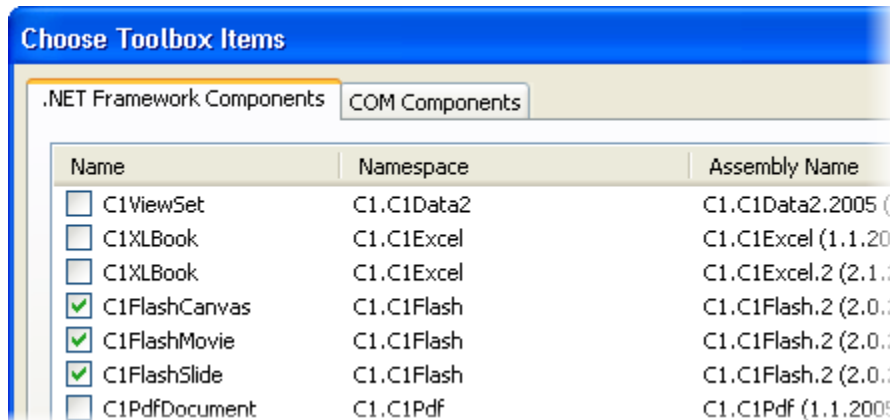
To manually add the **Flash for .NET** controls to the Visual Studio Toolbox:

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu, if necessary) and right-click it to open the context menu.
2. To make the **C1Flash** components appear on their own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **C1Flash**, for example.



3. Right-click the tab where the components are to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.

4. In the dialog box, go to the **.NET Framework Components** tab. Sort the list by Namespace (click the Namespace column header) and check the check box for the component belonging to namespace **C1.C1Flash**. Note that there may be more than one component for each namespace.



Adding Flash for .NET Components to the Form

To add **ComponentOne Flash for .NET** components to a form:

1. Add the **Flash for .NET** controls to the Visual Studio Toolbox.
2. Double-click the control or drag it onto your form.

Adding a Reference to the Assembly

To add a reference to the **C1Flash** assembly:

1. Select the **Add Reference** option from the **Project** menu of your project.
2. Select the **ComponentOne C1Flash** assembly from the list on the **.NET** tab or browse to find the C1.C1Flash.2.dll file and click **OK**.
3. Double-click the form caption area to open the code window. At the top of the file, add the following **Imports** statements (**using** in C#):
`Imports C1.C1Flash`

Note: This makes the objects defined in the **C1Flash** assembly visible to the project. See [Namespaces](#) (page 12) for more information.

Migrating a Flash for .NET Project to Visual Studio 2005

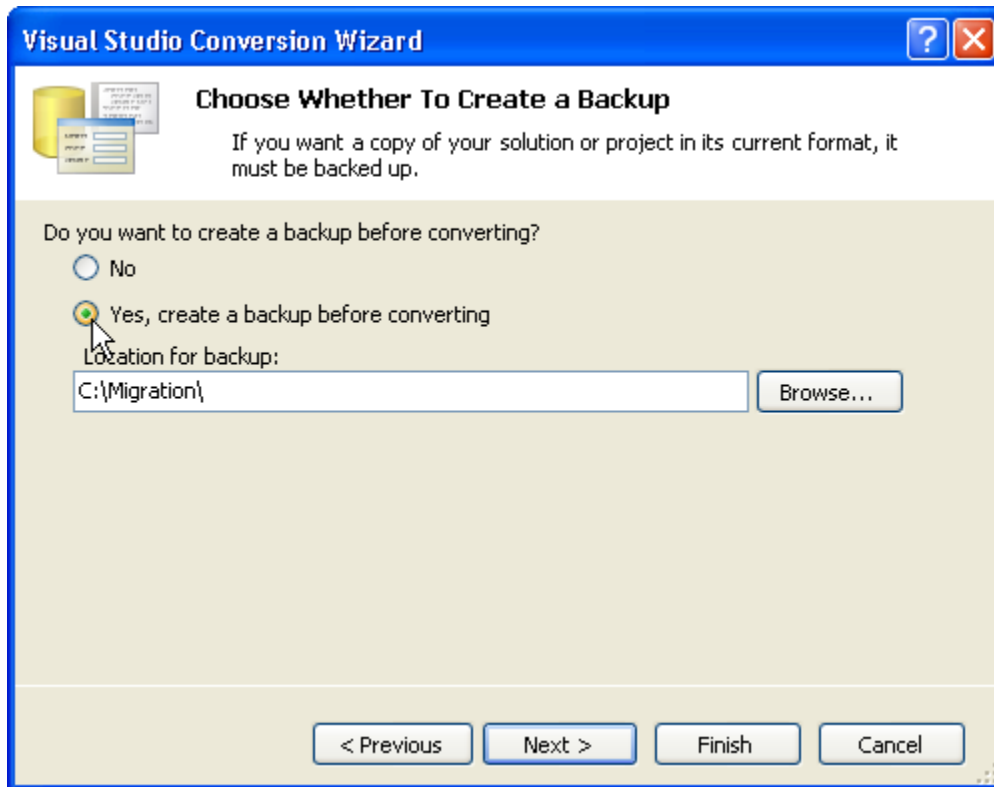
To migrate a project using ComponentOne components to Visual Studio 2005, there are two main steps that must be performed. First, you must convert your project to Visual Studio 2005, which includes removing any references to a previous assembly and adding a reference to the new assembly. Secondly, the .licx file, or licensing file, must be updated in order for the project to run correctly.

To convert the project:

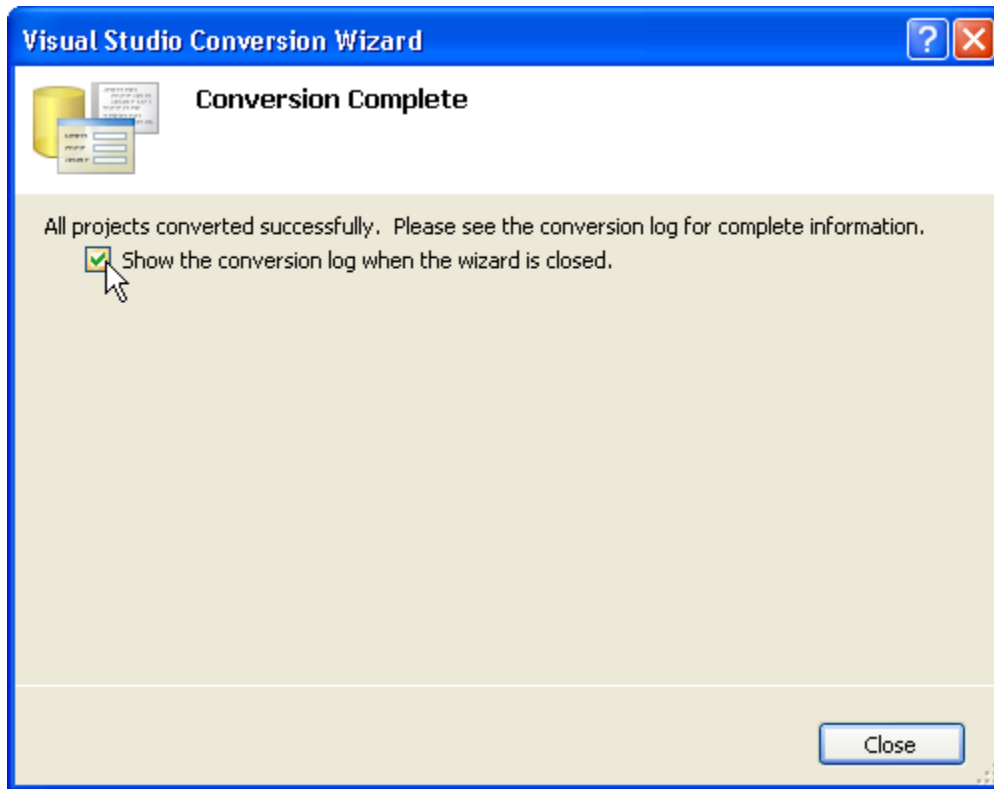
1. Open Visual Studio 2005 and select **File | Open | Project/Solution**.
2. Locate the **.sln** file for the project that you wish to convert to Visual Studio 2005. Select it and click **Open**. The **Visual Studio Conversion Wizard** appears.



3. Click **Next**.
4. Select **Yes, create a backup before converting** to create a backup of your current project and click **Next**.

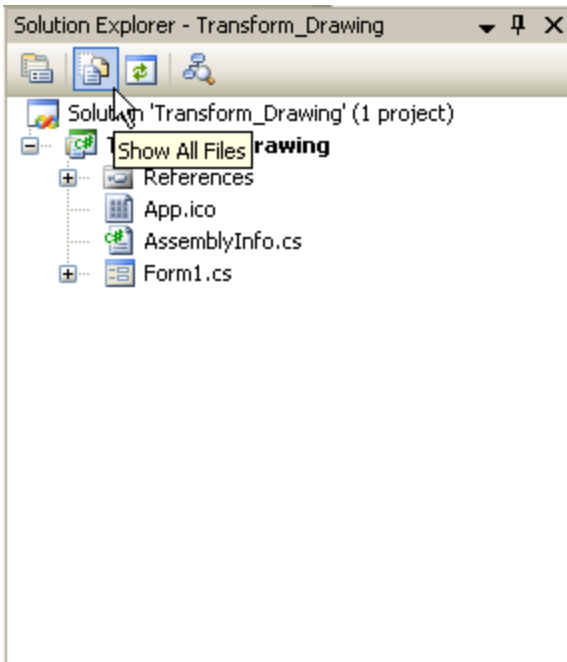


5. Click **Finish** to convert your project to Visual Studio 2005. The **Conversion Complete** window appears.
6. Click Show the conversion log when the wizard is closed if you want to view the conversion log.



7. Click **Close**. The project opens. Now you must remove references to any of the previous ComponentOne .dlls and add references to the new ones.
8. Go to the Solution Explorer (**View | Solution Explorer**) and click the **Show All Files** button.

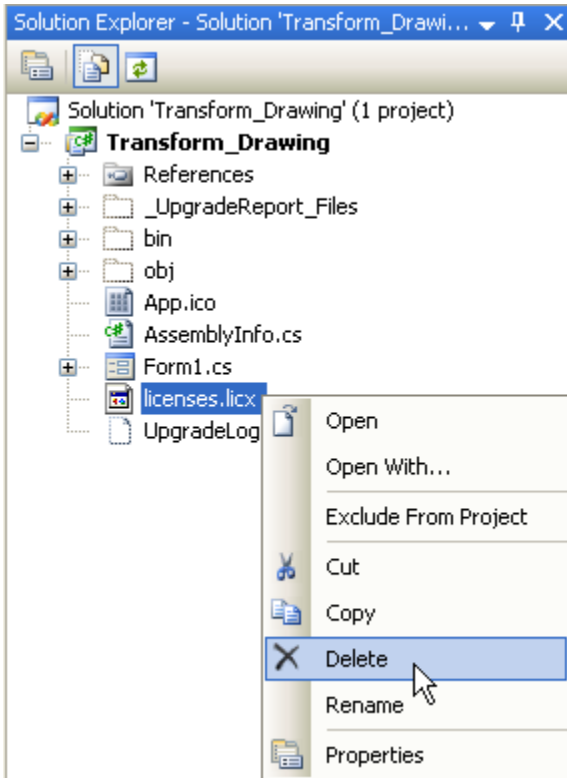
Note: The **Show All Files** button does not appear in the Solution Explorer toolbar if the Solution project node is selected.



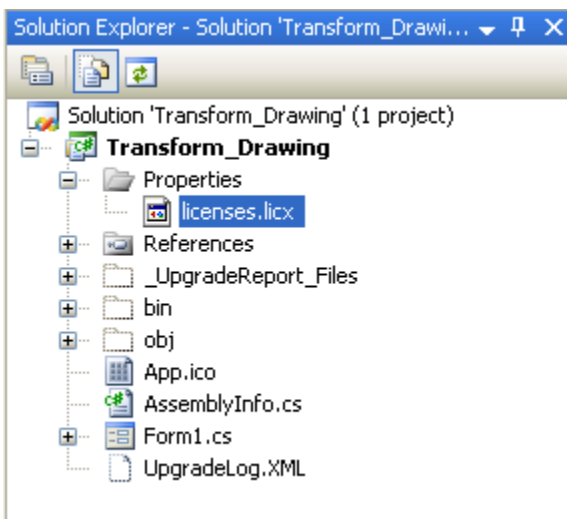
9. Expand the **References** node, right-click C1.C1Flash, and select **Remove**.
10. Right-click the **References** node and select **Add Reference**.
11. Locate and select **C1.C1Flash.2.dll**. Click **OK** to add it to the project.

To update the .licx file:

1. In the Solution Explorer, right-click the **licenses.licx** file and select **Delete**.



2. Click **OK** to permanently delete **licenses.licx**. The project must be rebuilt to create a new, updated version of the .licx file.
3. Click the **Start Debugging** button to compile and run the project. The new .licx file may not be visible in the Solution Explorer.
4. Select **File | Close** to close the form and then double-click the **Form.vb** or **Form.cs** file in the Solution Explorer to reopen it. The new **licenses.licx** file appears in the list of files.



The migration process is complete.

Key Features

Create colorful Adobe Flash files. Benefit from using **ComponentOne Flash for .NET**, including:

- **The ability to create dynamic Adobe Flash files in your application** – Create Flash files at run time. Link the output file inside a Web page to instantly display the dynamic Flash content.
- **Powerful drawing abilities** – Provides most of the graphical drawing abilities that Adobe Flash (SWF) format supports – anti-aliasing, fast rendering to a bitmap of any color format, and animation.

The Adobe Flash file format (SWF) delivers vector graphics and animation over the Internet to the Adobe Flash Player. The SWF file format is designed to be a very efficient delivery format, not a format for exchanging graphics between graphics editors. It is designed to meet the following goals – goals with which some other file formats cannot compete:

- **On-screen display:** The format is primarily intended for on-screen display and supports anti-aliasing, fast rendering to a bitmap of any color format, animation.
- **Extensibility:** The format is a tagged format, so it can be evolved with new features while maintaining backward compatibility with earlier versions of Flash Player.
- **Network delivery:** The format can travel over a network with limited and unpredictable bandwidth.
- **Simplicity:** The format is simple so that Flash Player is small and easily ported.
- **Scalability:** The files work well on limited hardware, and can take advantage of better hardware when it is available. This is important because computers have different monitor resolutions and bit depths.
- **Speed:** The files render at a high quality very quickly.

For more information on the Flash format and Flash utilities from Adobe and other sources, visit the [Adobe Web site](#).

- **An object model that parallels the .NET Graphics class** – C1FlashCanvas methods and properties for drawing graphics are the exact same as those available in the .NET Graphics class. You do not need to know the mechanism; C1Flash describes the graphical content, and how the complex SWF tags are organized and written on a file stream are totally transparent to the user. Discover how quickly you can learn C1Flash.
- **Highly-secure text** – Text in Flash format provides higher security to users since it is difficult to copy.
- **The ability to render any .NET Image object** – Render any regular .NET Image object into a SWF file: metafiles are re-played and simulated into Flash graphics. Use C1Flash to transfer a metafile format into a Flash format.
- **C1FlashSlide Designer available to quickly create Flash slides** – Design the layout and content of slide shows in WYSIWYG design surface with zero code.

Create Canvas Documents with C1FlashCanvas

The following topics demonstrate how to use the C1FlashCanvas component to create a Flash document and add text, graphics, and images.

Creating Canvas Documents

To create a single frame Adobe Flash document using `C1FlashCanvas`, the following three steps are required:

1. Create a `C1FlashCanvas` object.
2. Add content to the document. This usually involves calling the **DrawXXX** or **FillXXX** methods.
3. Render the document to a file or to a stream using the corresponding methods.

For more information on how to create canvas documents, see the [C1FlashCanvas Tasks](#) (page 44) topic.

To follow tradition, here's how to create a "hello world" document using `C1FlashCanvas`:

- Visual Basic

```
' step 1: create the C1FlashCanvas object
Dim canvas As New C1FlashCanvas()

' step 2: add content to the page
Dim rc = New Rectangle(20, 20, 200, 40)
Dim font As New Font("Arial", 12)
canvas.DrawString("Hello World!", font, Brushes.Black, rc)

' step 3: save the document to a file
canvas.RenderToFile("")Dim world As hello
```
- C#

```
// step 1: create the C1FlashCanvas object
C1FlashCanvas canvas = new C1FlashCanvas();

// step 2: add content to the page
RectangleF rc = new Rectangle(20, 20, 200, 40);
Font font = new Font("Arial", 12);
canvas.DrawString("Hello World!", font, Brushes.Black, rc);

// step 3: save the document to a file
canvas.RenderToFile(@"c:\temp\hello world.swf");
```

Step 2 is the most interesting one. The code starts by creating a new rectangle, then creates a **Font** object and calls the `DrawString` method to write "Hello World!" on the canvas. This is exactly what you would do if you were writing to a **Graphics** object in .NET, and is what makes **Flash for .NET** so easy to use.

One important thing to remember is that `C1FlashCanvas` uses a logical pixel coordinate system with the origin at the top-left corner of the page. This is similar to the default coordinate system used by .NET.

Adding Text to C1FlashCanvas

This topic demonstrates how to add text to your Flash document.

Drawing Text

Adding text to `C1FlashCanvas` is easy; all the work is done by the `DrawString` method.

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

```
canvas.DrawString("Hello World!", font, Brushes.Black, rc);
```

By default, `DrawString` will align the text to the left and to the top of the given rectangle and will wrap the string within the rectangle. You can change these options by specifying a *StringFormat* parameter in the call to `DrawString`. The *StringFormat* has members that allow you to specify the horizontal alignment (**Alignment**), vertical alignment (**LineAlignment**), and flags that control wrapping and clipping.

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

- **Visual Basic**

```
Dim font As New Font("Arial", 12)
Dim rc As New RectangleF(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 = StringAlignment.Center

canvas.DrawString([text], font, Brushes.Black, rc, sf)
canvas.DrawRectangle(Pens.Gray, rc)
```
- **C#**

```
Font font = new Font("Arial", 12);
RectangleF rc = new RectangleF(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 = StringAlignment.Center;

canvas.DrawString(text, font, Brushes.Black, rc, sf);
canvas.DrawRectangle(Pens.Gray, rc);
```

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 **MeasureString** method for that. **MeasureString** returns a **SizeF** structure that contains the width and height of the string (in points) when rendered with a given font.

Drawing HTML Text

DrawString provides all the functionality you need for rendering paragraphs using a single font and color.

You can also use **DrawStringHtml** to render a limited subset of the HTML tag language with a few additions not normally present in HTML. The following tags are supported:

Tag	Description
<p> ... </p>	Defines a paragraph. The attribute align may be present, with value left, right, or center.
 	Inserts a line break.
<a> ... 	Defines a hyperlink. The attribute href must be present. The attribute target is optional, and specifies a window name.
 ... 	Defines a span of text that uses a given font. The following attributes are available: <ul style="list-style-type: none">• face, which specifies a font name that must match a font name supplied in a DefineFont2 tag• size, which is specified in twips, and may include a leading '+' or '-' for

	relative sizes <ul style="list-style-type: none"> • color, which is specified as a #RRGGBB hex triplet
 ... 	Defines a span of bold text.
<i> ... </i>	Defines a span of italic text.
<u> ... </u>	Defines a span of underlined text.
 ... 	Defines a bulleted paragraph. The tag is not necessary and is not supported. Numbered lists are not supported.
<textformat> ... </textformat>	Defines a span of text with certain formatting options. The following attributes are available: <ul style="list-style-type: none"> • leftmargin, which specifies the left margin in twips • rightmargin, which specifies the right margin in twips • indent, which specifies the left indent in twips • blockindent, which specifies a block indent in twips • leading, which specifies the leading in twips • tabstops, which specifies a comma-separated list of tab stops, each specified in twips
<tab>	Inserts a tab character, which advances to the next tab stop as defined with <textformat>

For example, the following code renders a line of text with some bold and italic characters in it, and the "Sample" is in green:

- Visual Basic

```
Dim font As New Font("Arial", 12)
Dim rect As New RectangleF(160, 120, 200, 60)
canvas.DrawStringHtml("<b><font color=\"#00FF00\">Sample</font></b>
<i>string</i>", font, Brushes.Red, rect)
```
- C#

```
Font font = new Font("Arial", 12);
Rectangle rect = new RectangleF(160, 120, 200, 60);
canvas.DrawStringHtml("<b><font color=\"#00FF00\">Sample</font></b>
<i>string</i>", font, Brushes.Red, rect);
```

Adding Images to C1FlashCanvas

Adding images to C1FlashCanvas is done by using the DrawImage method.

DrawImage draws a given image at a specified location and with its original size or given size. For example, the following code loads a bitmap from resource and draws the image at a position:

- Visual Basic

```
Dim a As [Assembly] = [Assembly].GetExecutingAssembly()
Dim an As String = a.GetName().Name
Dim bmp As New Bitmap(a.GetManifestResourceStream((an + ".lvhover.jpg")))
Dim c1logo As New Bitmap(a.GetManifestResourceStream((an + ".c1logo.jpg")))
canvas.DrawImage(c1logo, New Point(320, 10))
```

```
canvas.DrawImage(c1logo, New Rectangle(10, 10, 200, 60))
```

- **C#**

```
Assembly a = Assembly.GetExecutingAssembly();  
string an = a.GetName().Name;  
Bitmap c1logo = new Bitmap(a.GetManifestResourceStream(an +  
".c1logo.jpg"));  
canvas.DrawImage(c1logo, new Point(320, 10));  
canvas.DrawImage(c1logo, new Rectangle(10, 10, 200, 60));
```

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 Flash document, metafiles are better than bitmap images.

Adding Graphics to C1FlashCanvas

The C1FlashCanvas 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.

The following example illustrates how similar the graphics methods are between C1FlashCanvas and the .NET **Graphics** class. The sample declares a C1FlashCanvas class and calls methods to draw/fill shapes with solid color, semi-transparent color, texture brush or gradient brush.

The point of the sample is that if you replaced the C1FlashCanvas class with a regular .NET **Graphics** object, you would be able to compile the code and get the same results:

- **Visual Basic**

```
Dim canvas As New C1FlashCanvas()  
  
'Draw line sections  
Dim pen As New Pen(Color.Red, 1)  
Dim points(4) As Point  
points(0) = New Point(40, 40)  
points(1) = New Point(150, 100)  
points(2) = New Point(150, 300)  
points(3) = New Point(300, 120)  
canvas.DrawLines(pen, points)  
  
' Draw Bezier curve  
Dim start As New Point(100, 100)  
Dim control1 As New Point(200, 10)  
Dim control2 As New Point(350, 50)  
Dim end1 As New Point(500, 100)  
Dim control3 As New Point(600, 150)  
Dim control4 As New Point(650, 250)  
Dim end2 As New Point(500, 300)  
Dim bezierPoints As Point() = {start, control1, control2, end1, control3,  
control4, end2}  
canvas.DrawBeziers(pen, bezierPoints)  
  
' Fill rectangle with solid color  
Dim rect As New Rectangle(200, 210, 120, 60)  
canvas.FillRectangle(Brushes.LightBlue, rect)  
  
' Fill the rectangle with color that has alpha value  
Dim c As Color = Color.FromArgb(90, Color.Blue)
```

```

Dim b As New SolidBrush(c)
rect.Offset(30, 30)
canvas.FillRectangle(b, rect)
b.Dispose()

' Create a texture brush
Dim a As [Assembly] = [Assembly].GetExecutingAssembly()
Dim an As String = a.GetName().Name
Dim bmp As New Bitmap(a.GetManifestResourceStream((an + ".lvhover.jpg")))
Dim tb As New TextureBrush(bmp)

' Fill the rectangle with texture brush
rect = New Rectangle(80, 120, 100, 120)
canvas.FillRectangle(tb, rect)

' Fill the pie with texture brush
rect = New Rectangle(300, 60, 150, 100)
canvas.FillPie(tb, rect, 30, 120)

' Draw a pie as border
pen.Color = Color.Green
pen.Width = 2
canvas.DrawPie(pen, rect, 30, 120)

' Create a linear gradient brush
Dim lb As New LinearGradientBrush(New Point(0, 0), New Point(100, 0),
Color.Blue, Color.Red)
Dim cb As New ColorBlend(3)
cb.Colors = New Color(3) {}
cb.Colors(1) = Color.Red
cb.Colors(2) = Color.Blue
cb.Colors(3) = Color.Yellow
cb.Positions = New Single(3) {}
cb.Positions(1) = 0
cb.Positions(2) = 0.5F
cb.Positions(3) = 1
lb.InterpolationColors = cb

' Fill the rectangle with the linear gradient brush
rect = New Rectangle(360, 200, 120, 40)
canvas.FillRectangle(lb, rect)

' Fill the ellipse with the linear gradient brush
rect = New Rectangle(360, 260, 120, 120)
canvas.FillEllipse(lb, rect)
lb.Dispose()

' Create a graphics path and add some graphical elements to this path
Dim graphPath As New GraphicsPath()
graphPath.AddEllipse(0, 0, 200, 100)
graphPath.AddRectangle(New Rectangle(20, 20, 200, 100))
graphPath.FillMode = FillMode.Winding
graphPath.AddString("Jason", FontFamily.GenericSansSerif, 1, 68, New
Rectangle(100, 320, 400, 100), StringFormat.GenericDefault)

' Fill the path
canvas.FillPath(Brushes.LightBlue, graphPath)

```

```

' Load the image from resource
Dim cllogo As New Bitmap(a.GetManifestResourceStream((an +
".cllogo.jpg")))

' Draw the image
canvas.DrawImage(cllogo, New Point(320, 10))
cllogo.Dispose()

' Draw some text
Dim font As New Font("MS Sans Serif", 15)
canvas.DrawString("Text in normal", font, Brushes.DarkOrange, New
PointF(20, 280))
font.Dispose()

' Draw some text in bold
font = New Font("MS Sans Serif", 15, FontStyle.Bold)
canvas.DrawString("Text in Bold", font, Brushes.DarkOrange, New PointF(20,
300))
font.Dispose()

' Draw some text in italic
font = New Font("MS Sans Serif", 15, FontStyle.Italic)
canvas.DrawString("Text in Italic", font, Brushes.DarkOrange, New
PointF(20, 320))

' Draw text with right alignment
rect = New Rectangle(20, 340, 150, 25)
canvas.DrawRectangle(Pens.Black, rect)
Dim sf As StringFormat = StringFormat.GenericDefault
sf.Alignment = System.Drawing.StringAlignment.Far
canvas.DrawString("Right alignment", font, Brushes.DarkOrange, rect, sf)
font.Dispose()
pen.Dispose()
tb.Dispose()
bmp.Dispose()

canvas.RenderToFile("")

```

- **C#**
C1FlashCanvas canvas = new C1FlashCanvas();

```

//Draw line sections
Pen pen = new Pen(Color.Red, 1);
Point[] points = new Point[4];
points[0] = new Point(40, 40);
points[1] = new Point(150, 100);
points[2] = new Point(150, 300);
points[3] = new Point(300, 120);
canvas.DrawLines(pen, points);

// Draw Bezier curve
Point start = new Point(100, 100);
Point control1 = new Point(200, 10);
Point control2 = new Point(350, 50);
Point end1 = new Point(500, 100);
Point control3 = new Point(600, 150);

```

```

Point control4 = new Point(650, 250);
Point end2 = new Point(500, 300);
Point[] bezierPoints = { start, control1, control2, end1, control3,
control4, end2 };
canvas.DrawBeziers(pen, bezierPoints);

// Fill rectangle with solid color
Rectangle rect = new Rectangle(200, 210, 120, 60);
canvas.FillRectangle(Brushes.LightBlue, rect);

// Fill the rectangle with color that has alpha value
Color c = Color.FromArgb(90, Color.Blue);
SolidBrush b = new SolidBrush(c);
rect.Offset(30, 30);
canvas.FillRectangle(b, rect);
b.Dispose();

// Create a texture brush
Assembly a = Assembly.GetExecutingAssembly();
string an = a.GetName().Name;
Bitmap bmp = new Bitmap(a.GetManifestResourceStream(an + ".lvhover.jpg"));
TextureBrush tb = new TextureBrush(bmp);

// Fill the rectangle with texture brush
rect = new Rectangle(80, 120, 100, 120);
canvas.FillRectangle(tb, rect);

// Fill the pie with texture brush
rect = new Rectangle(300, 60, 150, 100);
canvas.FillPie(tb, rect, 30, 120);

// Draw a pie as border
pen.Color = Color.Green;
pen.Width = 2;
canvas.DrawPie(pen, rect, 30, 120);

// Create a linear gradient brush
LinearGradientBrush lb = new LinearGradientBrush(new Point(0, 0), new
Point(100, 0), Color.Blue, Color.Red);
ColorBlend cb = new ColorBlend(3);
cb.Colors = new Color[3];
cb.Colors[1] = Color.Red;
cb.Colors[2] = Color.Blue;
cb.Colors[3] = Color.Yellow;
cb.Positions = new float[3];
cb.Positions[1] = 0;
cb.Positions[2] = 0.5F;
cb.Positions[3] = 1;
lb.InterpolationColors = cb;

// Fill the rectangle with the linear gradient brush
rect = new Rectangle(360, 200, 120, 40);
canvas.FillRectangle(lb, rect);

// Fill the ellipse with the linear gradient brush
rect = new Rectangle(360, 260, 120, 120);
canvas.FillEllipse(lb, rect);

```

```

lb.Dispose();

// Create a graphics path and add some graphical elements to this path
GraphicsPath graphPath = new GraphicsPath();
graphPath.AddEllipse(0, 0, 200, 100);
graphPath.AddRectangle(new Rectangle(20, 20, 200, 100));
graphPath.FillMode = FillMode.Winding;
graphPath.AddString("Jason", FontFamily.GenericSansSerif, 1, 68, new
Rectangle(100, 320, 400, 100), StringFormat.GenericDefault);

// Fill the path
canvas.FillPath(Brushes.LightBlue, graphPath);

// Load the image from resource
Bitmap cllogo = new Bitmap(a.GetManifestResourceStream(an +
".cllogo.jpg"));

// Draw the image
canvas.DrawImage(cllogo, new Point(320, 10));
cllogo.Dispose();

// Draw some text
Font font = new Font("MS Sans Serif", 15);
canvas.DrawString("Text in normal", font, Brushes.DarkOrange, new
PointF(20, 280));
font.Dispose();

// Draw some text in bold
font = new Font("MS Sans Serif", 15, FontStyle.Bold);
canvas.DrawString("Text in Bold", font, Brushes.DarkOrange, new PointF(20,
300));
font.Dispose();

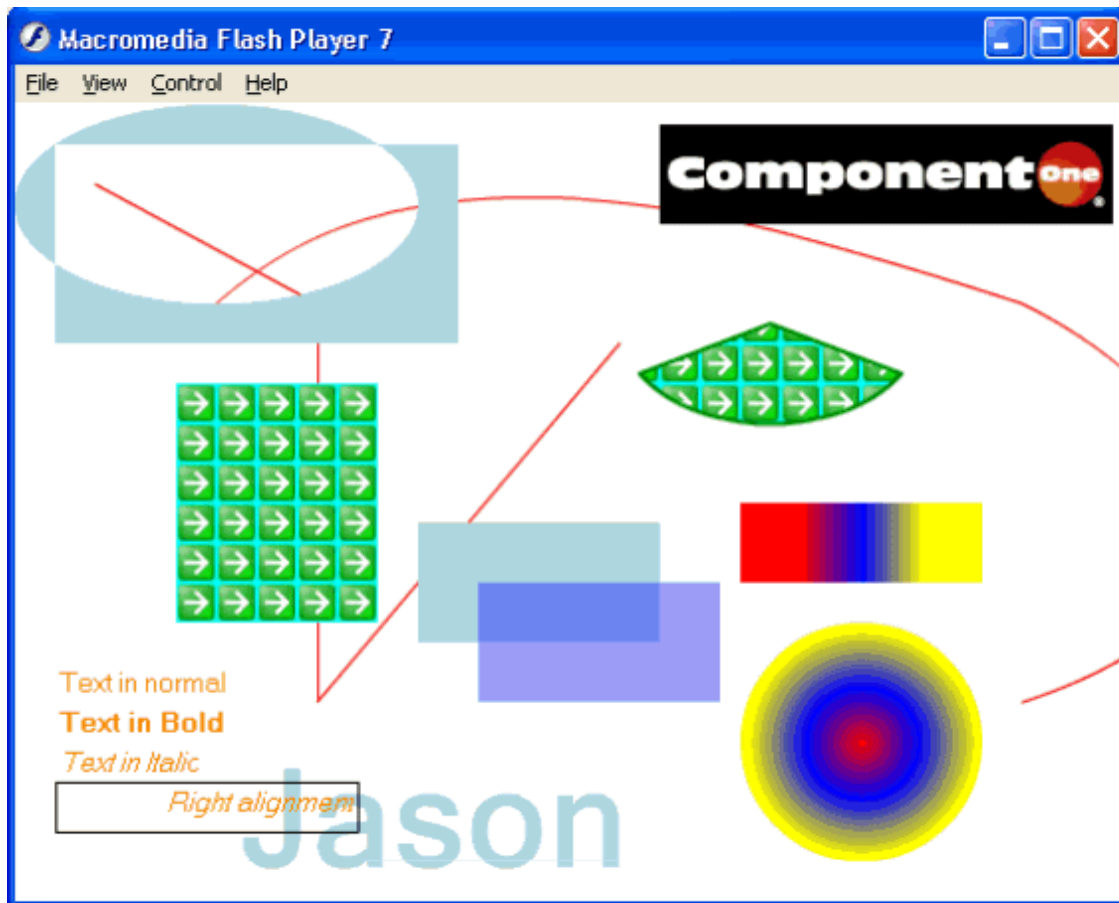
// Draw some text in italic
font = new Font("MS Sans Serif", 15, FontStyle.Italic);
canvas.DrawString("Text in Italic", font, Brushes.DarkOrange, new
PointF(20, 320));

// Draw text with right alignment
rect = new Rectangle(20, 340, 150, 25);
canvas.DrawRectangle(Pens.Black, rect);
StringFormat sf = StringFormat.GenericDefault;
sf.Alignment = System.Drawing.StringAlignment.Far;
canvas.DrawString("Right alignment", font, Brushes.DarkOrange, rect, sf);
font.Dispose();
pen.Dispose();
tb.Dispose();
bmp.Dispose();

canvas.RenderToFile(@"c:\temp\gdi.swf");

```

Here is the resulting Flash document:



Transforming the Graphic

As in the .NET **Graphics** class, you can rotate/scale/translate the coordinate by modifying the Transform property or call the corresponding methods of the C1FlashCanvas.

The following example rotates and scales the coordinate in a circle and draws the same rectangle in each coordinate.

- Visual Basic


```
Dim canvas As New C1FlashCanvas()

' Resets the coordinate transform
canvas.ResetTransform()
Dim ptCenter As New Point(canvas.Width / 2, canvas.Height / 2)

' Moves the coordinate origin point to the center of the canvas
canvas.TranslateTransform(ptCenter.X, ptCenter.Y)
Dim rect As New Rectangle(0, 0, 100, 40)
Dim i As Integer
For i = 0 To 11

' Draws the rectangle with the same rectangle parameter
canvas.DrawRectangle(Pens.Orange, rect)

' Rotates the coordination by 30 degrees
canvas.RotateTransform(30)
```

```
        ' Scales the coordination
        canvas.ScaleTransform(1.075F, 1.075F)
    Next i

    canvas.RenderToFile("c:\temp\transform.swf")
```

- **C#**

```
C1FlashCanvas canvas = new C1FlashCanvas();

// Resets the coordinate transform
canvas.ResetTransform();
Point ptCenter = new Point(canvas.Width/2, canvas.Height/2);

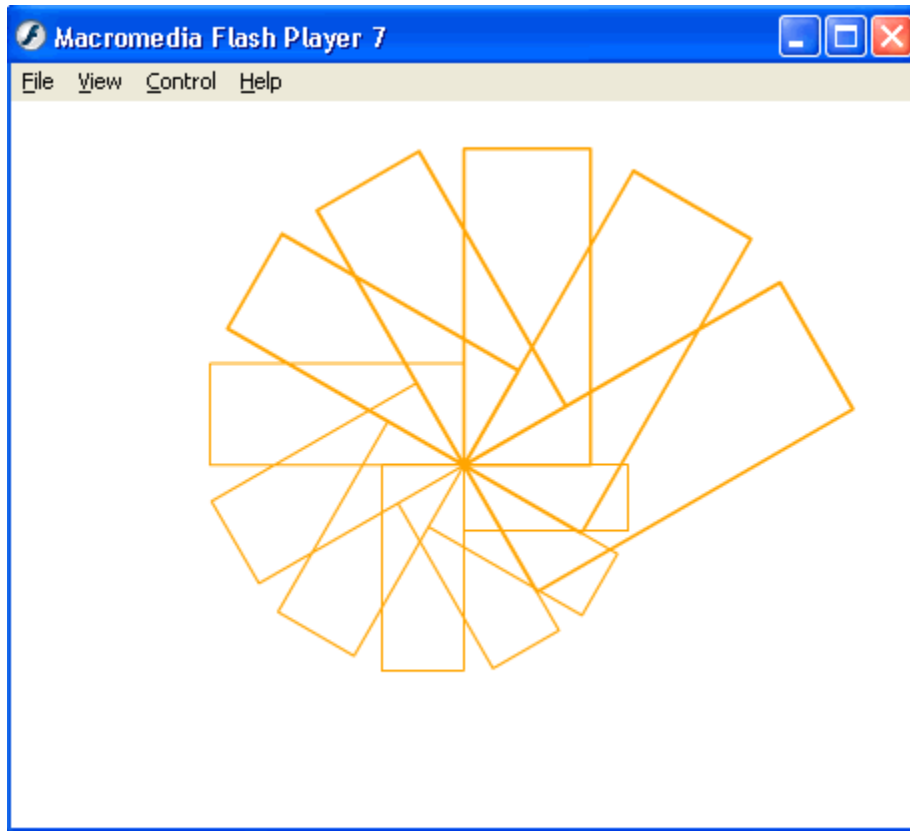
// Moves the coordinate origin point to the center of the canvas
canvas.TranslateTransform(ptCenter.X, ptCenter.Y);
Rectangle rect = new Rectangle(0, 0, 100, 40);
for(int i = 0; i < 12; i++)
{
    // Draws the rectangle with the same rectangle parameter
    canvas.DrawRectangle(Pens.Orange, rect);

    // Rotates the coordination by 30 degrees
    canvas.RotateTransform(30);

    // Scales the coordination
    canvas.ScaleTransform(1.075F, 1.075F);
}

canvas.RenderToFile(@"c:\temp\transform.swf");
```

Here is the result of the code:



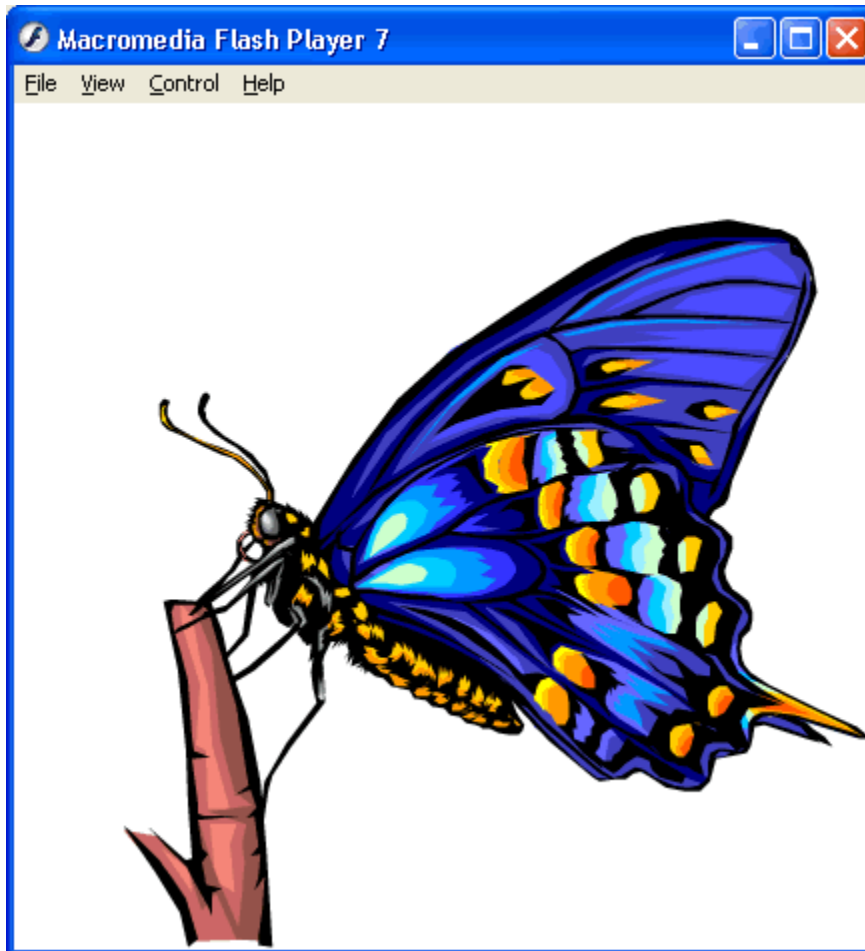
Using Metafiles to Render Graphics

C1FlashCanvas 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 C1FlashCanvas. 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 C1FlashCanvas using the DrawImage command. This method allows you to expose any graphics you create as images or as Flash documents.

The metafiles could be generated by a reporting engine, drawing or charting program, or any application that can create metafile images.

Here is a result of replaying a metafile in Flash document:



Create Movie Documents with C1FlashMovie

The following topics explain how a Flash movie is composed and show you how to use the C1FlashMovie component to create your own Flash movie.

Understanding Frames and Graphical Objects

A flash animation is composed of a series of frames. Each frame is displayed and then promptly replaced another frame to create the illusion of movement. This same technique is used to create television shows, motion pictures, and traditional cell animation.

Users are allowed to add and remove graphical objects to frames. There are many graphical objects defined in **ComponentOne Flash for .NET**, including **FLine**, **FRectangle**, **FCircle**, **FOval**, **FArc**, **FText**, **FEditText**, **FPolygon**, **FBeziars**, and **FPath**. Each object represents a basic graphical element or shape that can be outlined or filled.

When a graphical object is added to one frame, it will be displayed in this frame and all of the following frames until it is removed. For example, to make a rotating animation, the graphical object must be added to a frame, then removed in the next frame, rotated, and added again to the new frame.

Frame 0



Create a FOval object, add it to Frame 0

Frame 1



Remove the object from Frame 1, rotate 45 degrees, and add it to Frame 1

Frame 2



Remove the object from Frame 2, rotate 90 degrees, and add it to Frame 2

Frame 3



Remove the object from Frame 3, rotate 135 degrees, and add it to Frame 3

Each graphical object needs to be assigned a depth value. The depth determines the stacking order of the object. Objects with lower depth values are displayed underneath objects with higher depth values. An object with a depth value of 1 is displayed at the bottom of the stack. An object may appear more than once in the frame but at different depths. There can be only one object at any given depth.

You do not need to create a frame; it will be created automatically when you access the frame index.

Note: **C1FlashMovie** uses *twip* as its coordinate measurement, which is the coordinate used by the Adobe SWF specification. In the SWF format, a *twip* is 1/20th of a logical pixel. A logical pixel is the same as a screen pixel when the movie is played at 100%—that is, without scaling.

Creating Movie Documents

Creating an Adobe Flash movie documents using C1FlashMovie requires the three following steps:

1. Create a C1FlashMovie object.
2. Create graphical objects and add/remove them to/from frames.
3. Render the document to a file or to a stream using the corresponding methods.

For more information on how to create movie documents, see the [C1FlashMovie Tasks](#) (page 58) topic.

The following graphic represents the oval object, frame by frame, created by the Flash movie:

Frame 0



Create a FOval object, add it to Frame 0

Frame 1



Remove the object from Frame 1, rotate 45 degrees, and add it

Frame 2



Remove the object from Frame 2, rotate 90 degrees, and add it

Frame 3



Remove the object from Frame 3, rotate 135 degrees, and add

to Frame 1

to Frame 2

it to Frame 3

The following example shows how to create a movie with a rotating oval as demonstrated in the above picture:

- Visual Basic

```
' Step 1: create the C1FlashMovie object
Dim movie As New C1FlashMovie()

' Step 2: add/remove graphical object to/from frames
' create an oval object
Dim rect As New Rectangle(100 * Constants.TWIPS, 100 * Constants.TWIPS,
200 * Constants.TWIPS, 100 * Constants.TWIPS)
Dim oval As New FOval(rect)

' set its out line color and width
oval.LineColor = Color.Red
oval.LineWidth = 2 * Constants.TWIPS
oval.Depth = System.Convert.ToUInt16(1)

' add to frame 0
movie.Frames(0).AddObject(oval)

' remove from frame 1, rotate 45 degrees, add it back
movie.Frames(1).RemoveObject(oval)
oval.Rotate(45F)
movie.Frames(1).AddObject(oval)

' remove from frame 2, rotate 90 degrees, add it back
movie.Frames(2).RemoveObject(oval)
oval.Rotate(90F)
movie.Frames(2).AddObject(oval)

' remove from frame 3, rotate 135 degrees, add it back
movie.Frames(3).RemoveObject(oval)
oval.Rotate(135F)
movie.Frames(3).AddObject(oval)

' Step 3: render to file
movie.RenderToFile("c:\temp\movie.swf")
LaunchViewer("c:\temp\movie.swf")
```

- C#

```
// Step 1: create the C1FlashMovie object
C1FlashMovie movie = new C1FlashMovie();

// Step 2: add/remove graphical object to/from frames
// create an oval object
Rectangle rect = new Rectangle( 100 * Constants.TWIPS, 100 *
Constants.TWIPS, 200 * Constants.TWIPS, 100 * Constants.TWIPS );
FOval oval = new FOval( rect );

// set its out line color and width
oval.LineColor = Color.Red;
oval.LineWidth = 2 * Constants.TWIPS;
oval.Depth = System.Convert.ToUInt16(1);
```

```

// add to frame 0
movie.Frames(0).AddObject( oval );

// remove from frame 1, rotate 45 degrees, add it back
movie.Frames(1).RemoveObject( oval );
oval.Rotate(45F);
movie.Frames(1).AddObject( oval );

// remove from frame 2, rotate 90 degrees, add it back
movie.Frames(2).RemoveObject( oval );
oval.Rotate(90F);
movie.Frames(2).AddObject( oval );

// remove from frame 3, rotate 135 degrees, add it back
movie.Frames(3).RemoveObject( oval );
oval.Rotate(135F);
movie.Frames(3).AddObject( oval );

// Step 3: render to file
movie.RenderToFile(@"c:\temp\movie.swf")
LaunchViewer(@"c:\temp\movie.swf");

```

One important thing to remember is that `C1FlashMovie` uses a *twip* coordinate system with the origin at the top-left corner of the page. This is different from the system used in the `C1FlashCanvas`, which is similar to what is used in the .NET framework. The `Constant` class defines the constant of *twip* per logical pixel, which is 20.

Create Slide Documents with C1FlashSlide

The following topics demonstrate how to use the `C1FlashSlide` component to create a slide document and modify its settings.

Creating Slide Documents

Creating a slide document in Adobe Slide format requires the following four steps:

1. Place a `C1FlashSlide` component on the form using a drag-and-drop operation.
2. Right-click the `C1FlashSlide` component and select **Design** from its pop-up menu. This specifies the layout and properties of the slide and other elements, like navigation buttons, page header, page footer and page number.
3. Add new pages and draw content to each page.
4. Render the document to a file or to a stream using the corresponding methods.

For more information on how to create slide documents, see the [C1FlashSlide Tasks](#) (page 63) topic.

Using the C1FlashSlide Designer

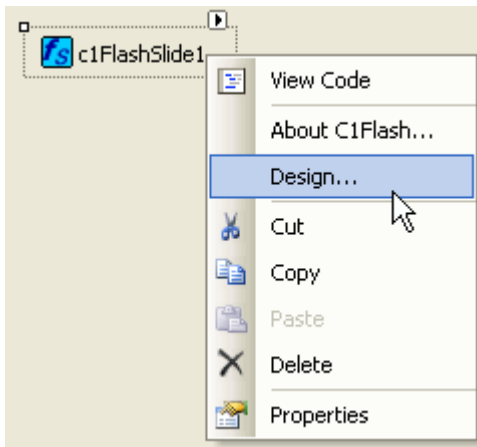
`C1FlashSlide Designer` is used to layout the built in UI elements. You can access the slide designer two ways:

- Click the smart tag (📌) located above the `C1FlashSlide` component and select **Design** from the `C1FlashSlide Tasks` menu.



OR

- Right-click the C1FlashSlide component and select **Design** from its pop-up menu.



After you have completed the slide design, your settings will be placed into the source code of the form and can be restored the next time you open the designer.

The intrinsic UI elements include:

- Page Header
- Page Footer
- Page Number
- Navigation Buttons

There are also four button elements used to navigate between pages:

- First Button – Click to navigate to the first page of the slide.
- Back Button – Click to navigate to the previous page.
- Next Button – Click to navigate to the next page.
- Last Button – Click to navigate to the last page of the slide.

To change the properties of each UI element, click the **Properties** tab on the right side and select a single or multiple elements. The properties of the element(s) will be displayed in the Property grid.

Page Header, Page Footer, and Page Number have the following properties:

- **Font** – The font of the UI element.
- **ForeColor** – The foreground color of the UI element.

- **Location** – The location of the UI element.
- **Size** – The size of the UI element.
- **Text** – The text displayed.
- **Visible** – The visibility of the UI element.

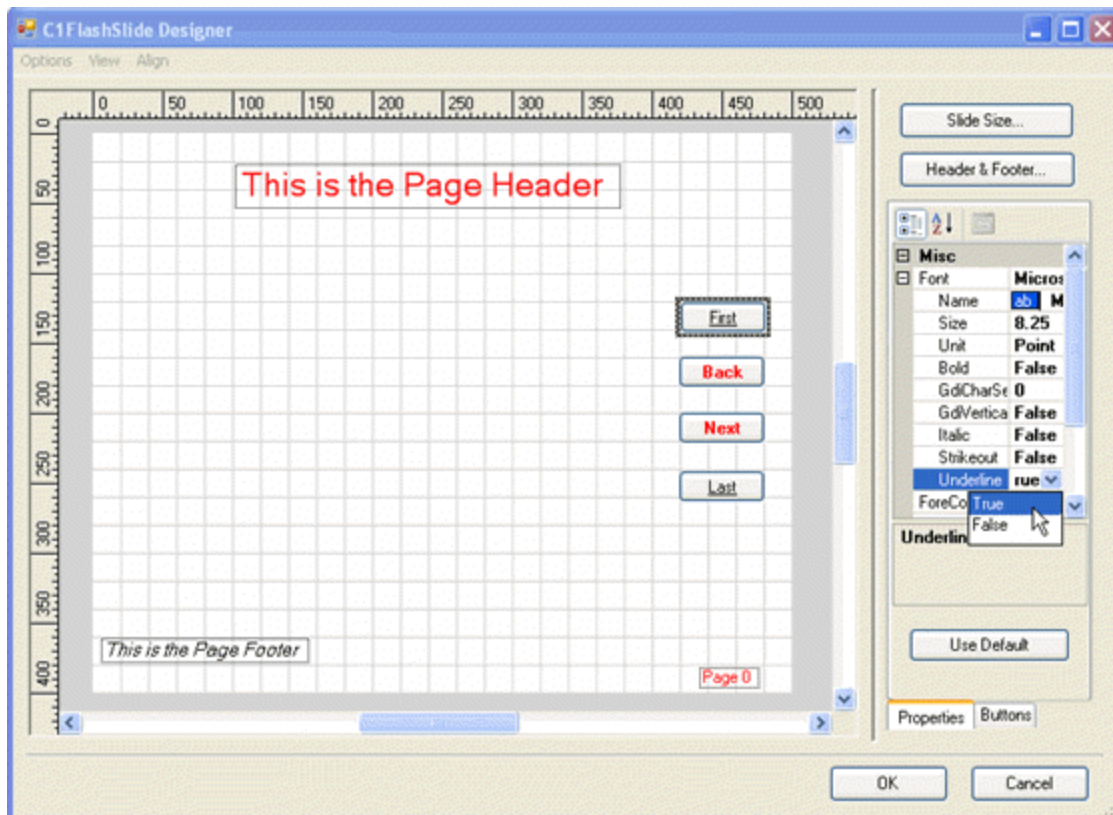
Button elements (for Normal, Arrow, and Mandarin styles) only have the following properties:

- **Location** – The location of the UI element.
- **Size** – The size of the UI element.
- **Visible** – The visibility of the UI element.

System button elements have the following properties:

- **Font** – The font of the UI element.
- **ForeColor** – The foreground color of the UI element.
- **Location** – The location of the UI element.
- **Size** – The size of the UI element.
- **Text** – The text displayed.
- **TextAlign** – The text alignment.
- **Visible** – The visibility of the UI element.

Here is an example of the **C1FlashSlide Designer** modified:



Note that the grid is available to help organize the objects on the slide. To view the grid, select **View | Show Grid** from the menu or right-click on the slide and select **Show Grid**.

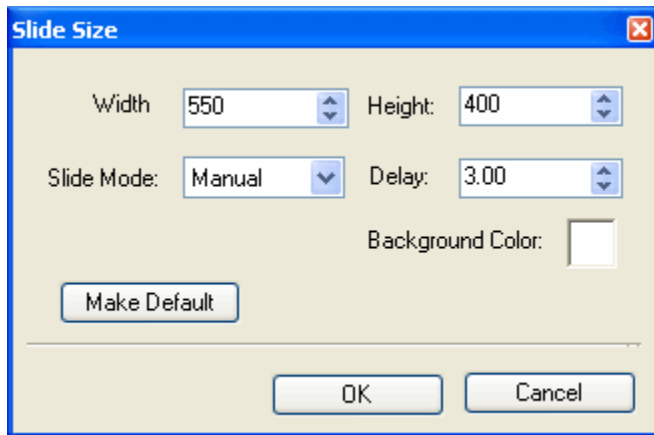
Setting Common Slide Attributes

The **Slide Size** dialog box allows you to change common attributes of the slide document. To access the **Slide Size** form, complete the following task:

- Click the **Slide Size** button.
- OR
- Select the **Options | Size** menu item.

The attributes include: **Width**, **Height**, **Slide Mode**, **Delay Time**, and **Background Color**.

Here is the **Slide Size** dialog box that shows the default settings:



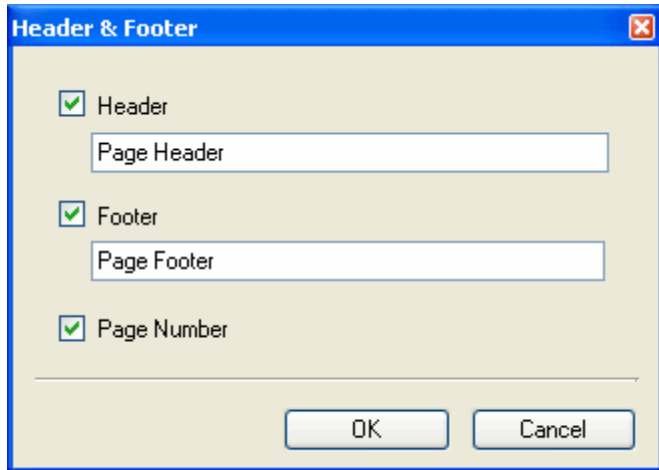
The **Slide Mode** can be set to one of the two values - *Manual* and *Automatic*.

When you set the **Slide Mode** to *Automatic*, the slide will be displayed automatically with the specified interval delay time. Also, no navigation button will be displayed even if the **Visible** property is set to **True**.

Setting the Header & Footer

The **Header & Footer** dialog box allows you to set the properties of **Page Header**, **Page Footer**, and **Page Number**. To access the **Header & Footer** dialog box, complete the following task:

- Click the **Header & Footer** button.
- OR
- Select the **Options | Header & Footer** menu item.



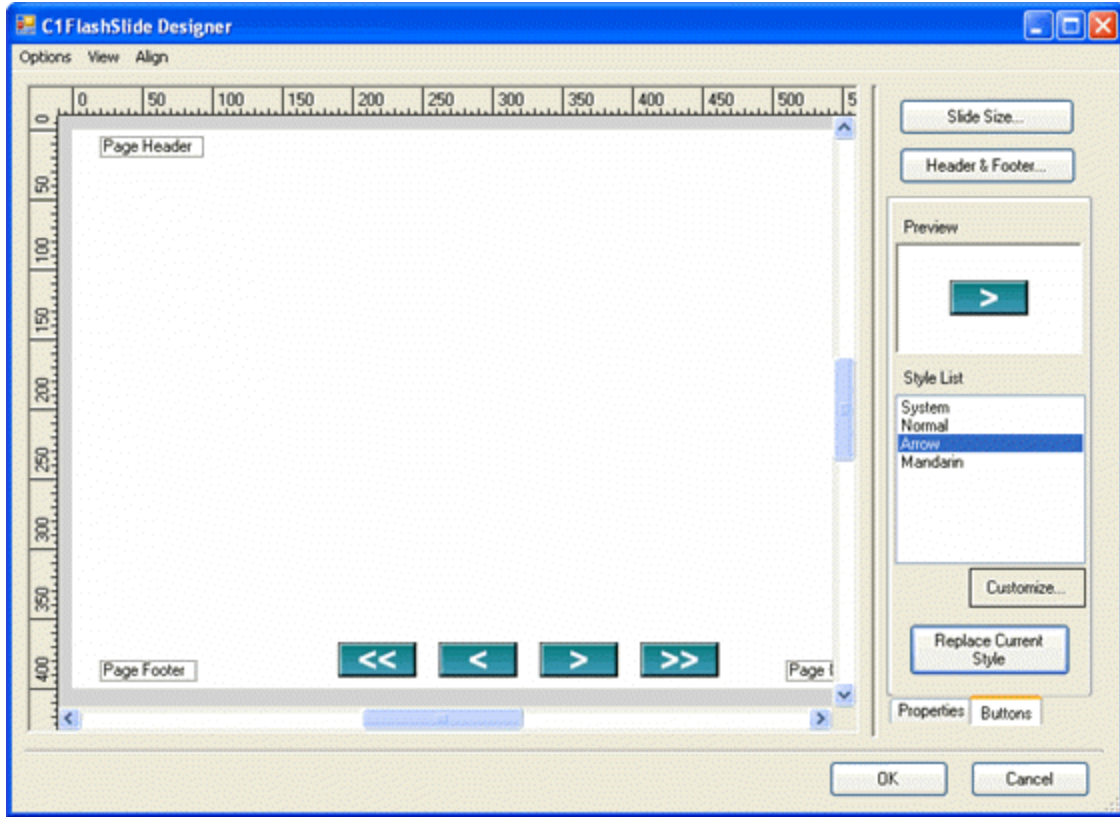
Setting the Button Style

CIFlashSlide Designer has sets of internally built button styles. To set the button style:

1. Click the **Buttons** tab in the lower-right corner.
2. Choose a satisfying button style from the style list.
3. Click the **Replace Current Style** button.

From the preview panel, you can preview the button style before making an actual replacement.

Here is an example using the **Arrow** button style:



Flash for .NET Samples

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

C# Samples

Sample	Description
Canvas_Bubbles	Demonstrates methods of drawing shapes or filling shapes. This sample uses the C1.C1Flash.C1FlashCanvas class.
Canvas_C1Chart	Chooses some of the demos from C1Chart , allows you to render the Chart in Meta file format and then reproduce the chart in SWF format. This sample uses the C1Chart and C1Chart3D controls.
Canvas_C1Image	Demonstrates how to render images using the C1FlashCanvas class. This sample uses the C1.C1Flash.C1FlashCanvas class.
Canvas_C1LineChart	Demonstrates the basic methods of drawing lines using the C1FlashCanvas class. This sample uses the C1.C1Flash.C1FlashCanvas class.
Canvas_C1Text	Demonstrates the methods of drawing text using the C1FlashCanvas class.
Canvas_C1Transform	Demonstrates the transformation of drawing using the C1FlashCanvas class.
Movie_Rotate	Demonstrates how to generate a movie using the C1FlashMovie class. This sample uses the C1.C1Flash.C1FlashMovie class.
Slide_Navigation	Demonstrates the methods of creating a slideshow that has navigation buttons. This

sample uses the C1.C1Flash.C1FlashSlide class.

Flash for .NET Task-Based Help

The task-based help assumes that you are familiar with programming in Visual Studio .NET environment. By following the steps outlined in the help, you will be able to utilize the features of **ComponentOne Flash for .NET**.

Each task-based help topic provides a solution for specific tasks using the **C1FlashCanvas**, **C1FlashMovie**, or **C1FlashSlide** component. Each topic also assumes that you have created a new .NET project. For additional information on this topic, see [Creating a .NET Project](#) (page 13).

C1FlashCanvas Tasks

The following topics assume that you have placed a **C1FlashCanvas** component on the form. For more information, see [Creating a .NET Project](#) (page 13).

To view the canvas in Flash Player:

Before you begin, you must create a **LaunchViewer** function to view the canvas in either Flash Player or Internet Explorer. Add the following code in the source file to create the function:

- Visual Basic

```
Private Sub LaunchViewer(ByVal filename As String)
    Try
        System.Diagnostics.Process.Start(filename)
    Catch
        System.Diagnostics.Process.Start("IEXPLORE.EXE", filename)
    End Try
End Sub
```
- C#

```
private void LaunchViewer(string filename)
{
    try
    {
        System.Diagnostics.Process.Start(filename);
    }
    catch (Exception e)
    {
        System.Diagnostics.Process.Start("IEXPLORE.EXE", filename);
    }
}
```

Drawing Text in C1FlashCanvas

The following topics demonstrate the methods of drawing text using the C1FlashCanvas class.

Drawing Text

To draw text on the C1FlashCanvas, use the RenderToFile method to render the content on the C1FlashCanvas to a SWF file:

1. To draw some simple text, such as **"Hello World"**, enter the following code in the **Form_Load** event:
 - Visual Basic

```
Imports C1.C1Flash
```

```

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Dim rc As New Rectangle(20, 20, 200, 40)
    Dim font As New Font("Arial", 14, FontStyle.Bold)
    Me.C1FlashCanvas1.DrawString("Hello World!", font, Brushes.Orange,
rc)
font.Dispose()
End Sub

```

- C#

```
using C1.C1Flash;
```

```

private void Form1_Load(object sender, System.EventArgs e)
{
    Rectangle rc = new Rectangle(20, 20, 200, 40);
    Font font = new Font("Arial", 14, FontStyle.Bold);
    this.c1FlashCanvas1.DrawString("Hello World!", font,
Brushes.Orange, rc);
font.Dispose();
}

```

2. Place a button on the form using a drag-and-drop operation and enter the following code in the **Button_Click** event, which will save the canvas to a SWF file and launch it to FlashPlayer or Internet Explorer:

- Visual Basic

```

Me.C1FlashCanvas1.RenderToFile("c:\WindowsApplication1.swf")
LaunchViewer("c:\WindowsApplication1.swf")

```

- C#

```

this.c1FlashCanvas1.RenderToFile(@"c:\WindowsApplication1.swf");
LaunchViewer(@"c:\WindowsApplication1.swf");

```

3. Save and run your application, then click the button.

This topic illustrates the following:

Here is what your text will look like in Internet Explorer:

Hello World!

Drawing Text Inside a Rectangle

To draw text inside of a rectangle, use the DrawRectangle method to draw a rectangle and call the DrawString method to draw text into the rectangle:

1. To place text in the rectangle, add the following code to the **Form_Load** event handler:

- Visual Basic

```
Imports C1.C1Flash
```

```

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Dim font As New Font("MS Sans Serif", 14)
    Dim rect As New Rectangle(50, 120, 200, 75)
    Me.C1FlashCanvas1.DrawRectangle(Pens.Black, rect)
    Me.C1FlashCanvas1.DrawString("Text in the rectangle", font,
Brushes.DarkOrange, rect)
font.Dispose()

```

```
End Sub
```

- C#
using Cl.C1Flash;

private void Form1_Load(object sender, System.EventArgs e)
{
 Font font = new Font("MS Sans Serif", 14);
 Rectangle rect = new Rectangle(50, 120, 200, 75);
 this.c1FlashCanvas1.DrawRectangle(Pens.Black, rect);
 this.c1FlashCanvas1.DrawString("Text in the rectangle", font,
Brushes.DarkOrange, rect);
 font.Dispose();
}

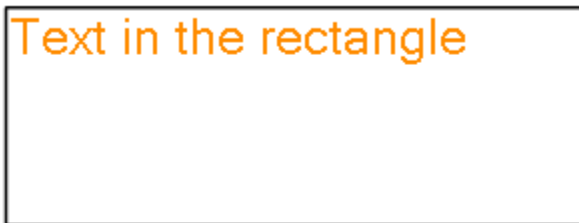
2. Add the following code to the **Button_Click** event, which will save the canvas to a SWF file and launch it to FlashPlayer or Internet Explorer:

- Visual Basic
Me.C1FlashCanvas1.RenderToFile("c:\WindowsApplication1.swf")
LaunchViewer("c:\WindowsApplication1.swf")
- C#
this.c1FlashCanvas1.RenderToFile(@"c:\WindowsApplication1.swf");
LaunchViewer(@"c:\WindowsApplication1.swf");

3. Save and run your application.

This topic illustrates the following:

Here is what your text will look like in Internet Explorer:



Drawing a Paragraph

To draw a paragraph within a rectangle region, complete the following tasks:

1. In the **Form_Load** event, enter the following code to draw a paragraph within a rectangle:

- Visual Basic
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
 Dim font As New Font("Courier New", 14, FontStyle.Regular)
 Dim rect As New Rectangle(20, 20, 300, 300)
 Dim s As String
 s = "ComponentOne was formed on July 1, 2000, by the merger of
APEX Software Corporation and VideoSoft. Both APEX and VideoSoft have a
history of leadership in the Microsoft Visual Studio component
industry."
 Me.C1FlashCanvas1.DrawString(s, font, Brushes.DarkRed, rect)
 font.Dispose()
End Sub

- C#


```
private void Form1_Load(object sender, System.EventArgs e)
{
    Rectangle rect = new Rectangle(20, 20, 300, 300);
    Font font = new Font("Courier New", 14, FontStyle.Regular);
    string s = "ComponentOne was formed on July 1, 2000, by the merger of
APEX Software Corporation and VideoSoft. Both APEX and VideoSoft have a
history of leadership in the Microsoft Visual Studio component
industry.";
    this.c1FlashCanvas1.DrawString(s, font, Brushes.DarkRed, rect);
    font.Dispose();
}
```

2. Enter the following code after the **font.Dispose** method in the **Form_Load** event to save the canvas to a SWF file and launch it in Internet Explorer:

- Visual Basic


```
Me.C1FlashCanvas1.RenderToFile("c:\c1flash_canvas_text.swf")
    LaunchViewer("c:\c1flash_canvas_text.swf")
```
- C#


```
this.C1FlashCanvas1.RenderToFile(@"c:\c1flash_canvas_text.swf");
    LaunchViewer(@"c:\c1flash_canvas_text.swf");
```

3. Save and run your application.

This topic illustrates the following:

Here is what your text will look like in Internet Explorer:

```
ComponentOne was formed on
July 1, 2000, by the merger
of APEX Software
Corporation and VideoSoft.
Both APEX and VideoSoft
have a history of
leadership in the Microsoft
Visual Studio component
industry.
```

Aligning Text in the Center

To draw text in the center, complete the following tasks:

1. To center text, add the following code to the **Form_Load** event:

- Visual Basic


```
Imports C1.C1Flash
```

```

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Dim font As New Font("Courier New", 14)
    Dim rect As New Rectangle(35, 170, 160, 35)
    Dim sf As New StringFormat(StringFormat.GenericDefault)
    rect.Offset(rect.Width + 30, 0)
    Me.C1FlashCanvas1.DrawRectangle(Pens.Black, rect)
    sf.Alignment = System.Drawing.StringAlignment.Center
    Me.C1FlashCanvas1.DrawString("Center alignment", font,
Brushes.DarkOrange, rect, sf)
End Sub

```

- C#

```
using C1.C1Flash;
```

```

private void Form1_Load(object sender, System.EventArgs e)
{
    Font font = new Font("MS Sans Serif", 14);
    Rectangle rect = new Rectangle(35, 170, 160, 35);
    StringFormat sf = new StringFormat(StringFormat.GenericDefault);
    rect.Offset(rect.Width + 30, 0);
    this.c1FlashCanvas1.DrawRectangle(Pens.Black, rect);
    sf.Alignment = System.Drawing.StringAlignment.Center;
    this.c1FlashCanvas1.DrawString("Center alignment", font,
Brushes.DarkOrange, rect, sf);
}

```

2. Add the following code to the **Button_Click** event, which will save the canvas to a SWF file and launch it to FlashPlayer or Internet Explorer:

- Visual Basic

```

Me.C1FlashCanvas1.RenderToFile("c:\WindowsApplication1.swf")
LaunchViewer("c:\WindowsApplication1.swf")

```

- C#

```

this.c1FlashCanvas1.RenderToFile(@"c:\WindowsApplication1.swf");
LaunchViewer(@"c:\WindowsApplication1.swf");

```

3. Save and run your application.

This topic illustrates the following:

Here is what your text will look like in Internet Explorer:

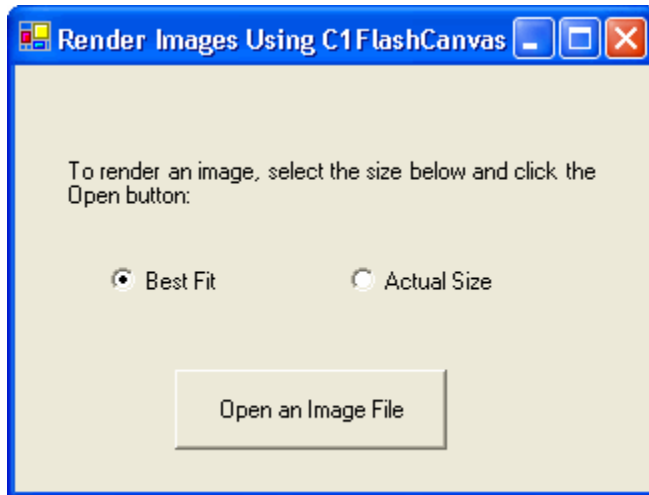


Rendering Images Using C1FlashCanvas

To render images using the C1FlashCanvas class, complete the following tasks:

Note: You can draw any image which is supported by the .NET framework onto the C1FlashCanvas, since the Flash only supports JPEG and BMP, **C1Flash** does the necessary conversion if not supported (specifically, JPEG, BMP, GIF, TIFF, PHG, ICON and WMF).

1. Place an **OpenFileDialog** control, **Button** and **Label** control, and two **RadioButton** controls to the form. Here is what the form will look like:



2. Add the following code to the **Button_Click** event handler:

- Visual Basic

```
Private Sub Button1_Click(sender As Object, e As System.EventArgs)
    Handles Button1.Click
    Me.OpenFileDialog1.InitialDirectory = Application.StartupPath
    Me.OpenFileDialog1.Filter =
        "All Image Files|*.png;*.emf;*.wmf;*.tif;*.tiff;*.gif;_
        *.jpg;*.jpe;*.jpeg;*.bmp;*.dib;*.rle" & _
        "|BMP (*.bmp;*.dib;*.rle)|*.bmp;*.dib;*.rle" & _
        "|JPEG (*.jpg;*.jpe;*.jpeg)|*.jpg;*.jpe;*.jpeg" & _
        "|WMF (*.wmf;*.emf)|*.wmf;*.emf" & _
        "|TIFF (*.tif;*.tiff)|*.tif;*.tiff" & _
        "|GIF (*.gif)|*.gif" & _
        "|PNG (*.png)|*.png" & _
        "|ICON (*.ico)|*.ico"

    If Me.OpenFileDialog1.ShowDialog() = DialogResult.OK Then '
        Dim filename As String = Me.OpenFileDialog1.FileName.Trim()
        If Not (filename Is Nothing) And filename.Length > 0 Then
            Dim image As Image = Image.FromFile(filename)
            Me.C1FlashCanvas1.Clear(Color.White)

            Dim width As Single = image.Width
            Dim height As Single = image.Height
            If Me.rbBestFit.Checked Then
                Dim ratio As Single = CSng(image.Width) /
                CSng(image.Height)
                If image.Width > image.Height Then
                    width = Me.C1FlashCanvas1.Width
```

```

        height = CInt(width / ratio)

        If height > Me.C1FlashCanvas1.Height Then
            height = Me.C1FlashCanvas1.Height
            width = height * ratio
        End If
    Else
        height = Me.C1FlashCanvas1.Height
        width = CInt(height * ratio)

        If width > Me.C1FlashCanvas1.Width Then
            width = Me.C1FlashCanvas1.Width
            height = width / ratio
        End If
    End If
End If

Dim x As Single = (Me.C1FlashCanvas1.Width - width) / 2
Dim y As Single = (Me.C1FlashCanvas1.Height - height) / 2

Me.C1FlashCanvas1.DrawImage(image, New RectangleF(x, y, width,
height))
    image.Dispose()
Me.C1FlashCanvas1.RenderToFile("c:\WindowsApplication1.swf")
    LaunchViewer("c:\WindowsApplication1.swf")
End If
End If
End Sub

```

- **C#**

```

private void button1_Click(object sender, System.EventArgs e)
{
    this.openFileDialog1.InitialDirectory = Application.StartupPath;
    this.openFileDialog1.Filter = "All Image
Files|*.png;*.emf;*.wmf;*.tif;*.tiff;*.gif;*.jpg;*.jpe;*.jpeg;*.bmp;*.dib;
*.rle|BMP(*.bmp;*.dib;*.rle)|*.bmp;*.dib;*.rle|JPEG(*.jpg;*.jpe;*.jpeg)
|*.jpg;*.jpe;*.jpeg|WMF(*.wmf;*.emf)|*.wmf;*.emf|TIFF(*.tif;*.tiff)|
*.tif;*.tiff|GIF(*.gif)|*.gif|PNG(*.png)|*.png|ICON(*.ico)|*.ico";
    if(this.openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string filename = this.openFileDialog1.FileName.Trim();
        if (filename != null && filename.Length > 0)
        {
            Image image = Image.FromFile(filename);
            this.c1FlashCanvas1.Clear(Color.White);

            float width = image.Width;
            float height = image.Height;
            if (this.rbBestFit.Checked)
            {
                float ratio = (float)image.Width/(float)image.Height;
                if (image.Width > image.Height)
                {
                    width = this.c1FlashCanvas1.Width;
                    height = (int)(width/ratio);

                    if (height > this.c1FlashCanvas1.Height)

```

```

        {
            height = this.clFlashCanvas1.Height;
            width = height * ratio;
        }
    }
    else
    {
        height = this.clFlashCanvas1.Height;
        width = (int)(height * ratio);

        if (width > this.clFlashCanvas1.Width)
        {
            width = this.clFlashCanvas1.Width;
            height = width / ratio;
        }
    }
}

float x = (this.clFlashCanvas1.Width - width)/2;
float y = (this.clFlashCanvas1.Height - height)/2;

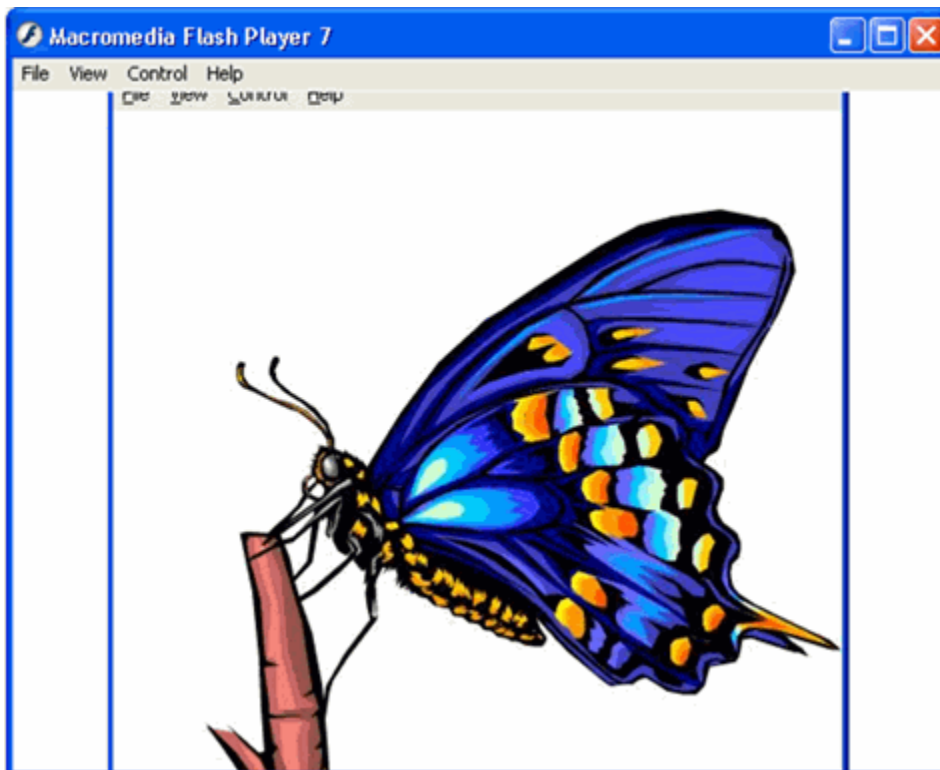
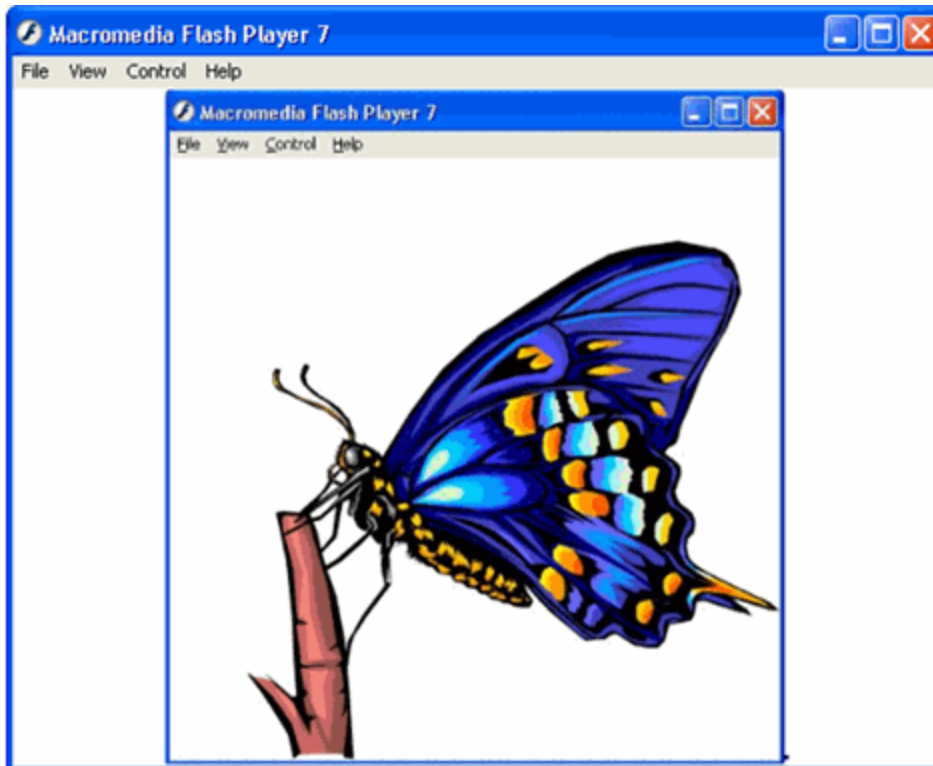
this.clFlashCanvas1.DrawImage(image, new RectangleF(x, y, width,
height));
image.Dispose();
this.clFlashCanvas1.RenderToFile(@"c:\WindowsApplication1.swf");
    LaunchViewer(@"c:\WindowsApplication1.swf");
}

```

3. Save and run your application.
4. Click the button.

This topic illustrates the following:

Here is what your chosen image (Best Fit and Actual Size, respectively) will look like:



Drawing Shapes or Filling Shapes in C1FlashCanvas

To draw an ellipse, use the DrawEllipse method and to fill the shape, call the FillEllipse method:

1. To begin drawing on the canvas, add the following code to the **Form_Load** event, which will draw an ellipse and fill the shape:

- Visual Basic

```
Imports Cl.C1Flash
```

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles MyBase.Load  
    Me.C1FlashCanvas1.Clear(Color.White)
```

```
    Dim rect As New Rectangle(100, 140, 140, 200)
```

```
        ' Draws the ellipse with the same ellipse parameter
```

```
        Me.C1FlashCanvas1.DrawEllipse(Pens.Red, rect)
```

```
        Me.C1FlashCanvas1.FillEllipse(Brushes.LightYellow, rect)
```

```
End Sub
```

- C#

```
using Cl.C1Flash;
```

```
private void Form1_Load(object sender, System.EventArgs e)
```

```
{
```

```
    this.c1FlashCanvas1.Clear(Color.White);
```

```
    Rectangle rect = new Rectangle(100, 140, 140, 200);
```

```
    // Draws the ellipse with the same ellipse parameter
```

```
    this.c1FlashCanvas1.DrawEllipse(Pens.Red, rect);
```

```
    this.c1FlashCanvas1.FillEllipse(Brushes.LightYellow, rect);
```

```
}
```

2. Enter the following code to the **Form_Load** event to save the canvas to a SWF file and launch it in Internet Explorer:

- Visual Basic

```
Me.C1FlashCanvas1.RenderToFile("c:\WindowsApplication1.swf")
```

```
    LaunchViewer("c:\WindowsApplication1.swf")
```

- C#

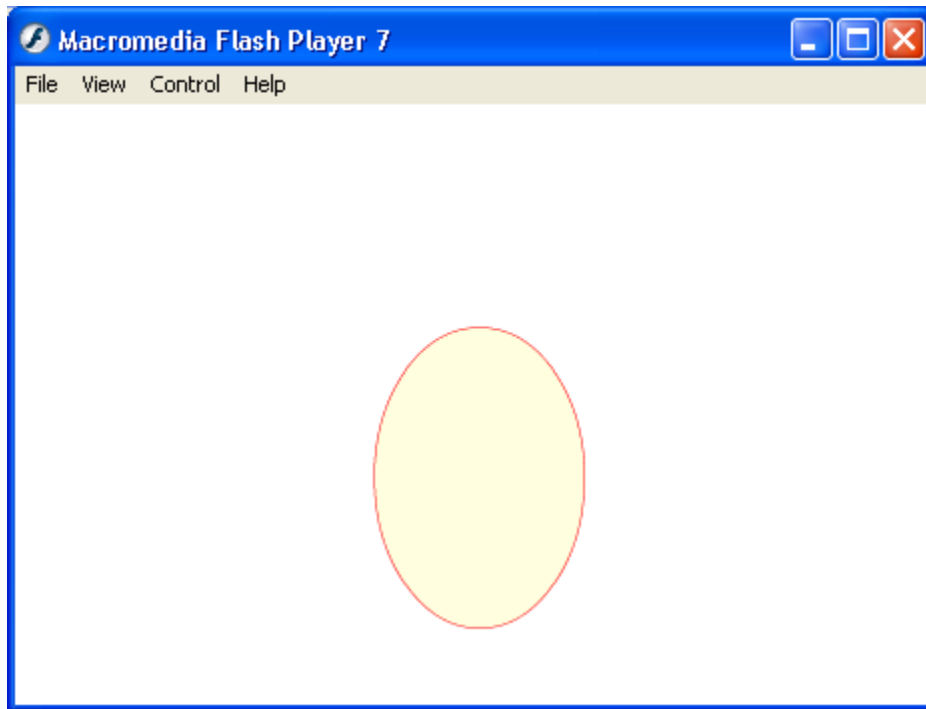
```
this.c1FlashCanvas1.RenderToFile(@"c:\WindowsApplication1.swf");
```

```
    LaunchViewer(@"c:\WindowsApplication1.swf");
```

3. Save and run your application.

This topic illustrates the following:

Here is what your drawing will look like:



Transforming a Drawing Using C1FlashCanvas

The following topics demonstrate the methods of transforming a drawing using the C1FlashCanvas class.

Rotating the Drawing

To draw an ellipse, use the DrawEllipse method and to transform the drawing, call the TranslateTransform and RotateTransform methods:

1. To draw an ellipse and transform it, add the following code to the **Form_Load** event:

- Visual Basic
Imports C1.C1Flash

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Me.C1FlashCanvas1.Clear(Color.White)

    ' Resets the coordinate transform
    Me.C1FlashCanvas1.ResetTransform()
    Dim ptCenter As New Point(Me.C1FlashCanvas1.Width / 2,
    Me.C1FlashCanvas1.Height / 20)

    ' Moves the coordination origin point to the center of the canvas
    Me.C1FlashCanvas1.TranslateTransform(ptCenter.X, ptCenter.Y)
    Dim rect As New Rectangle(80, 140, 140, 180)
    Dim i As Integer
    i = 0
    For i = 1 To 11

        ' Draws the ellipse with the same ellipse parameter
        Me.C1FlashCanvas1.DrawEllipse(Pens.Red, rect)
        Me.C1FlashCanvas1.FillEllipse(Brushes.LightYellow, rect)
    Next i
End Sub
```

```

        ' Rotates the coordination by 15 degrees
        Me.C1FlashCanvas1.RotateTransform(15)
    Next i
End Sub

```

- **C#**
using C1.C1Flash;
- ```

private void Form1_Load(object sender, System.EventArgs e)
{
 this.c1FlashCanvas1.Clear(Color.White);

 // Resets the coordinate transform
 this.c1FlashCanvas1.ResetTransform();
 Point ptCenter = new Point(this.c1FlashCanvas1.Width / 2,
this.c1FlashCanvas1.Height / 20);

 // Moves the coordination origin point to the center of the canvas
 this.c1FlashCanvas1.TranslateTransform(ptCenter.X, ptCenter.Y);
 Rectangle rect = new Rectangle(80, 140, 140, 180);
 for(int i = 0; i < 12; i++)

 {
 // Draws the ellipse with the same ellipse parameter
 this.c1FlashCanvas1.DrawEllipse(Pens.Red, rect);
 this.c1FlashCanvas1.FillEllipse(Brushes.LightYellow, rect);

 // Rotates the coordination by 15 degrees
 this.c1FlashCanvas1.RotateTransform(15);
 }
}

```

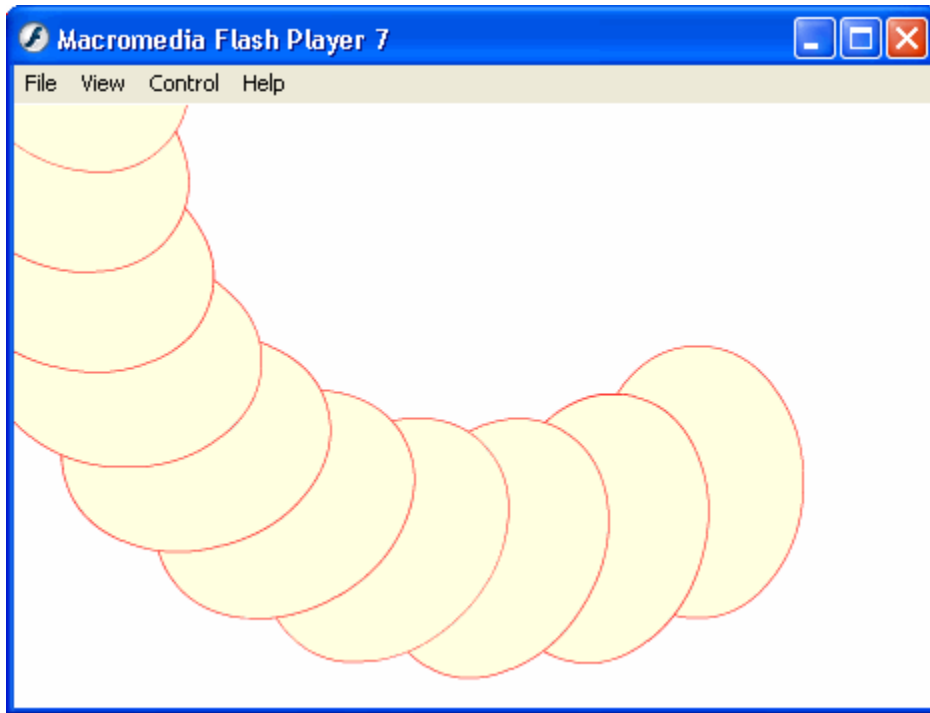
2. Enter the following code to the **Form\_Load** event to save the canvas to a SWF file and launch it to FlashPlayer or Internet Explorer:

- **Visual Basic**  
Me.C1FlashCanvas1.RenderToFile("c:\WindowsApplication1.swf")  
LaunchViewer("c:\WindowsApplication1.swf")
- **C#**  
this.c1FlashCanvas1.RenderToFile(@"c:\WindowsApplication1.swf");  
LaunchViewer(@"c:\WindowsApplication1.swf");

3. Save and run your application.

**This topic illustrates the following:**

Here is what your drawing will look like in Internet Explorer:



### ***Scaling the Coordination***

These instructions assume that you have already completed the [Rotating the Drawing](#) (page 54) task.

To draw an ellipse, use the DrawEllipse method and to scale the coordination, call the ScaleTransform method:

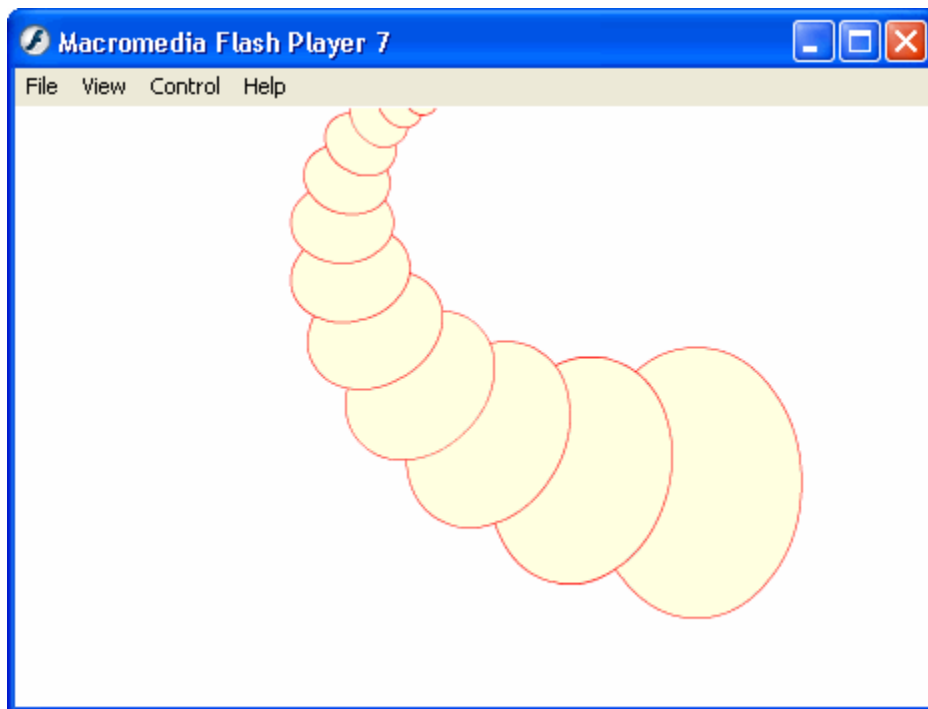
1. Find the RotateTransform method in the **Form\_Load** event and enter the following code beneath it:

- Visual Basic  
`Me.C1FlashCanvas1.ScaleTransform(0.85F, 0.85F)`
- C#  
`this.c1FlashCanvas1.ScaleTransform(0.85F, 0.85F);`

2. Run your application, then save the canvas to a SWF file and launch it in Internet Explorer.

**This topic illustrates the following:**

Your scaled drawing should resemble the following image:



### ***Transforming the Drawing***

These instructions assume that you have completed the [Scaling the Coordination](#) (page 56) task.

This topic uses the `DrawEllipse` method to draw an ellipse and calls the `ResetTransform` method to reset the transform space. A new transform space is specified by setting the `Transform` property.

Before you reset the transformation and multiply the drawing, import the **Drawing2D** and **C1Flash** namespaces; add the following code to the top of the form:

- Visual Basic  
`Imports System.Drawing.Drawing2D`  
`Imports C1.C1Flash`
- C#  
`using System.Drawing.Drawing2D;`  
`using C1.C1Flash;`

In the **Form\_Load** event handler, you will add code to reset the transformation and multiply the drawing. Complete the following steps:

1. Add the following code to the **Form\_Load** event, placing it *after* the code that sets the **DrawEllipse**, **FillEllipse**, **RotateTransform**, and **ScaleTransform** properties but *before* the **RenderToFile** method:

- Visual Basic  
`' Resets the coordinate`  
`Me.C1FlashCanvas1.ResetTransform()`  
  
`Dim i As Integer`  
`For i = 1 To 11`  
`Me.C1FlashCanvas1.DrawEllipse(Pens.Red, rect)`  
  
`' Change the Transform`  
`Dim m As Matrix = Me.C1FlashCanvas1.Transform`

```

 m.Shear(0.15F, 0.15F)
 Me.C1FlashCanvas1.Transform = m
 Next i

```

- C#
 

```

// Resets the coordinate
this.c1FlashCanvas1.ResetTransform();

for(int i = 1; i < 12; i++)
{
 this.c1FlashCanvas1.DrawEllipse(Pens.Red, rect);

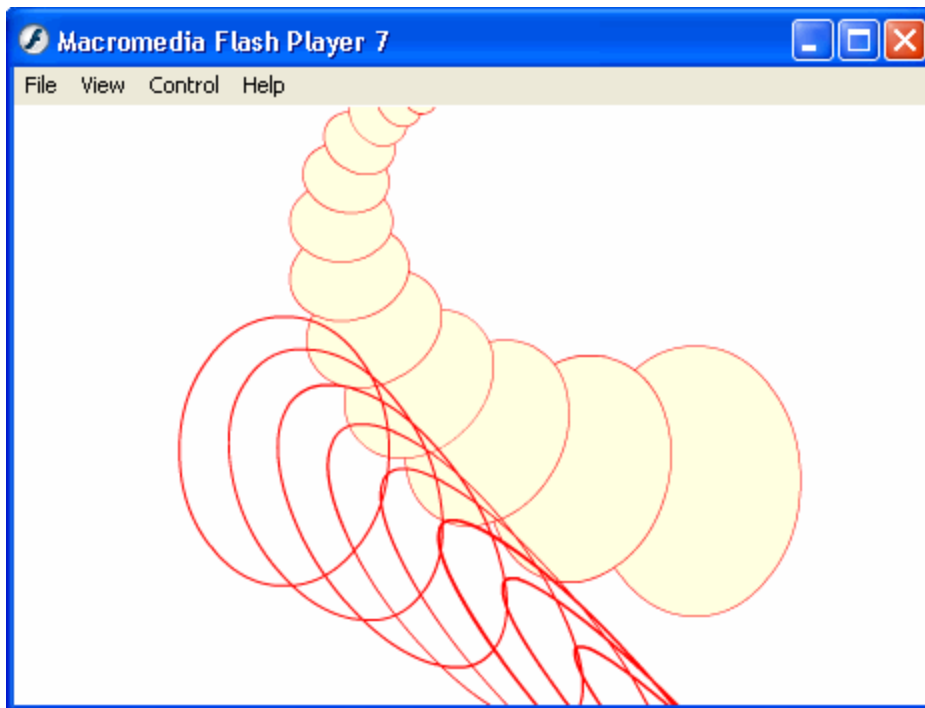
 // Change the Transform
 Matrix m = this.c1FlashCanvas1.Transform;
 m.Shear(0.15F, 0.15F);
 this.c1FlashCanvas1.Transform = m;
}

```

2. Save and run the application.

**This topic illustrates the following:**

Your transformed matrix drawing will look like the following image:



## C1FlashMovie Tasks

The following topics assume that you have placed a C1FlashMovie component on the form. For more information, see [Creating a .NET Project](#) (page 13).

Before you begin you must create a **LaunchViewer** function to view the movie in Flash Player, if available, or Internet Explorer. Add the following code in the source file to create the function:

- Visual Basic
 

```
Imports C1.C1Flash

Private Sub LaunchViewer(ByVal filename As String)
 Try
 System.Diagnostics.Process.Start(filename)
 Catch
 System.Diagnostics.Process.Start("IEXPLORE.EXE", filename)
 End Try
End Sub
```
- C#
 

```
using C1.C1Flash;

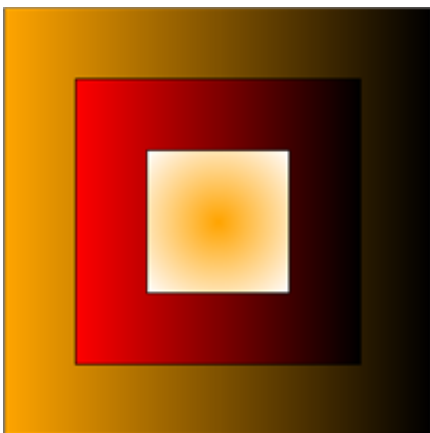
private void LaunchViewer(string filename)
{
 try
 {
 System.Diagnostics.Process.Start(filename);
 }
 catch (Exception e)
 {
 System.Diagnostics.Process.Start("IEXPLORE.EXE", filename);
 }
}
```

### Creating Movie Documents that Rotate

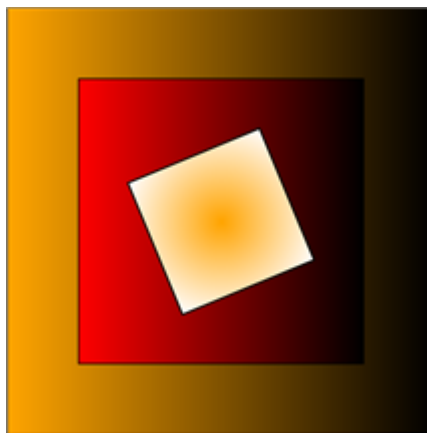
This topic demonstrates how to generate a movie using the C1FlashMovie class.

After completing the following tasks, run your application, save the canvas to a SWF file, and launch it in Internet Explorer. Here is a frame-by-frame representation of the animation that will appear in Internet Explorer:

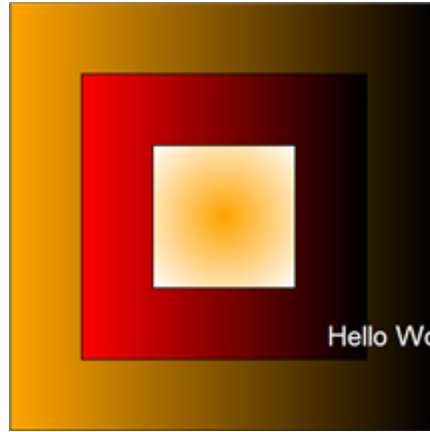
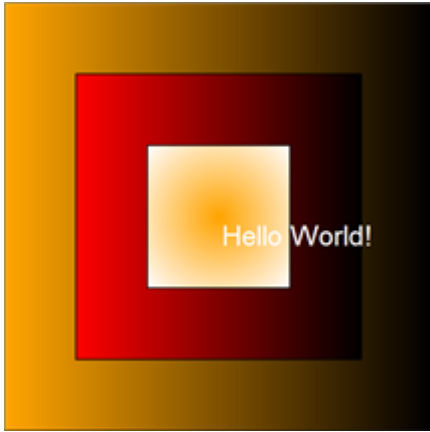
**Rectangles 1 (orange/black), 2 (red/black), Rotate Rectangle 3 and 3 (white/orange)**



**Text appears**



**Text falls off to the lower-right**



### Determining Each Rectangle's Location

To determine the location of each rectangle, add the following code to the **Form\_Load** event:

- Visual Basic
 

```
' top-left rectangle
Dim rect1 As New FRectangle(0 * Constants.TWIPS, 0 * Constants.TWIPS, 300
* Constants.TWIPS, 300 * Constants.TWIPS)
' middle rectangle
Dim rect2 As New FRectangle(50 * Constants.TWIPS, 50 * Constants.TWIPS,
200 * Constants.TWIPS, 200 * Constants.TWIPS)
' bottom-right rectangle
Dim rect3 As New FRectangle(100 * Constants.TWIPS, 100 * Constants.TWIPS,
100 * Constants.TWIPS, 100 * Constants.TWIPS)
```
- C#
 

```
// top-left rectangle
FRectangle rect1 = new FRectangle(0 * Constants.TWIPS, 0 *
Constants.TWIPS, 300 * Constants.TWIPS, 300 * Constants.TWIPS);
// middle rectangle
FRectangle rect2 = new FRectangle(50 * Constants.TWIPS, 50 *
Constants.TWIPS, 200 * Constants.TWIPS, 200 * Constants.TWIPS);
// bottom-right rectangle
FRectangle rect3 = new FRectangle(100 * Constants.TWIPS, 100 *
Constants.TWIPS, 100 * Constants.TWIPS, 100 * Constants.TWIPS);
```

### Filling Each Rectangle and Determining its Depth

To fill each rectangle and determine the depth of each rectangle, call the `FillGradientColor` method and `Depth` property. Add the following code to the **Form\_Load** event:

- Visual Basic
 

```
' set rect1 to radial-fill orange-black
rect1.FillGradientColor(Color.Orange, Color.Black, True)

' set rect3 to radial-fill red-black
rect2.FillGradientColor(Color.Red, Color.Black, True)

' set rect3 to radial-fill orange-white
rect3.FillGradientColor(Color.Orange, Color.White, False)

' set rect1 in depth 1 (at bottom)
rect1.Depth = 1
' set rect2 in depth 2 (in the middle)
```

```
rect2.Depth = 2
' set rect3 in depth 3 (on top of all)
rect3.Depth = 3
```

- **C#**

```
// set rect1 to radial-fill orange-black
rect1.FillGradientColor(Color.Orange, Color.Black, true);

// set rect3 to radial-fill red-black
rect2.FillGradientColor(Color.Red, Color.Black, true);

// set rect3 to radial-fill orange-white
rect3.FillGradientColor(Color.Orange, Color.White, false);

// set rect1 in depth 1 (at bottom)
rect1.Depth = 1;
// set rect2 in depth 2 (in the middle)
rect2.Depth = 2;
// set rect3 in depth 3 (on top of all)
rect3.Depth = 3;
```

### Creating Frames and Rotating a Rectangle

To place rectangles 1 and 2 on frame one, and make rectangle 3 rotate by itself, use the `AddObject` and `RemoveObject` methods. Add the following code to the **Form\_Load** event:

- **Visual Basic**

```
' "add" rect1 and rect2 to frame1(this.ClFlashMovie1 dictionary), then
"place" them on frame1
Me.ClFlashMovie1.Frames(0).AddObject(rect1)
Me.ClFlashMovie1.Frames(0).AddObject(rect2)
Me.ClFlashMovie1.Frames(0).AddObject(rect3)

' rect3 is rotating by itself.
Me.ClFlashMovie1.Frames(1).RemoveObject(rect3)
rect3.Rotate(22.5F)
Me.ClFlashMovie1.Frames(1).AddObject(rect3)
Me.ClFlashMovie1.Frames(2).RemoveObject(rect3)
rect3.Rotate(45F)
Me.ClFlashMovie1.Frames(2).AddObject(rect3)
Me.ClFlashMovie1.Frames(3).RemoveObject(rect3)
rect3.Rotate(67.5F)
Me.ClFlashMovie1.Frames(3).AddObject(rect3)
Me.ClFlashMovie1.Frames(4).RemoveObject(rect3)
rect3.Rotate(90F)
Me.ClFlashMovie1.Frames(4).AddObject(rect3)
```
- **C#**

```
// "add" rect1 and rect2 to frame1(this.clFlashMovie1 dictionary), then
"place" them on frame1
this.clFlashMovie1.Frames(0).AddObject(rect1);
this.clFlashMovie1.Frames(0).AddObject(rect2);
this.clFlashMovie1.Frames(0).AddObject(rect3);

// rect3 is rotating by itself
this.clFlashMovie1.Frames(1).RemoveObject(rect3);
rect3.Rotate(22.5F);
this.clFlashMovie1.Frames(1).AddObject(rect3);
this.clFlashMovie1.Frames(2).RemoveObject(rect3);
```

```

rect3.Rotate(45F);
this.clFlashMovie1.Frames(2).AddObject(rect3);
this.clFlashMovie1.Frames(3).RemoveObject(rect3);
rect3.Rotate(67.5F);
this.clFlashMovie1.Frames(3).AddObject(rect3);
this.clFlashMovie1.Frames(4).RemoveObject(rect3);
rect3.Rotate(90F);
this.clFlashMovie1.Frames(4).AddObject(rect3);

```

## Adding Moving Text

To add moving text, call the `FEditText` constructor. Add the following code in the **Form\_Load** event handler:

- Visual Basic

```

Dim [text] As New FEditText(New Rectangle(100 * Constants.TWIPS, 100 *
Constants.TWIPS, 300 * Constants.TWIPS, 80 * Constants.TWIPS), "Hello
World!", "Arial", False, False)
[text].ReadOnly = True
[text].NoSelect = True
[text].Height = 20 * Constants.TWIPS
[text].AutoSize = True
[text].ForeColor = Color.SkyBlue
Dim i As Integer
For i = 5 To 29
 [text].ForeColor = Color.White
 [text].Depth = CType(i, System.UInt16)
 [text].Translate(i * 10 * Constants.TWIPS, i * 10 * Constants.TWIPS)
 Me.ClFlashMovie1.Frames(i).AddObject([text])

 ' Remove the frame later in the timeline. Note you need to remove it
now,
 ' since every time the AddObject is called, the object Id changes. But
 ' you can remove in the future, which is what is happening here.
 If i < 30 Then
 Me.ClFlashMovie1.Frames((i + 1)).RemoveObject([text])
 End If
Next i

```

- C#

```

FEditText text = new FEditText(new Rectangle(100 * Constants.TWIPS, 100 *
Constants.TWIPS,
300 * Constants.TWIPS, 80 * Constants.TWIPS), "ComponentOne Hello World!",
"Arial", false, false);
text.ReadOnly = true;
text.NoSelect = true;
text.Height = 20 * Constants.TWIPS;
text.AutoSize = true;
text.ForeColor = Color.SkyBlue;
for(int i = 5; i < 30; i++)
{
 text.ForeColor = Color.White;
 text.Depth = (ushort)i;
 text.Translate(i * 10 * Constants.TWIPS, i * 10 * Constants.TWIPS);
 this.clFlashMovie1.Frames(i).AddObject(text);

 // Remove the frame later in the timeline. Note you need to remove it
now,
 // since every time the AddObject is called, the object Id changes. But

```

```

// you can remove in the future, which is what is happening here.
if (i < 30)
 this.c1FlashMovie1.Frames(i+1).RemoveObject(text);
}

```

### Saving and Running Your Application

1. Enter the following code in the **Form\_Load** event handler to save the canvas to a SWF file and launch it to FlashPlayer or Internet Explorer:
  - Visual Basic

```

Me.C1FlashMovie1.RenderToFile("c:\WindowsApplication1.swf")
LaunchViewer("C:\WindowsApplication1.swf")

```
  - C#

```

this.c1FlashMovie1.RenderToFile(@"c:\WindowsApplication1.swf");
LaunchViewer(@"C:\WindowsApplication1.swf");

```
2. Save and run your application.

## C1FlashSlide Tasks

The following topic assumes that you have placed a C1FlashSlide component on the form. For more information, see [Creating a .NET Project](#) (page 13).

By using the C1FlashSlide component, you can render multiple pages as an *automatic* or *user-controlled* slideshow. By default, the **SlideMode** is *Manual*. You can change the **SlideMode** to *Automatic* in C1FlashSlide properties pane or with the **C1FlashSlide Designer**, which is available by right-clicking the C1FlashSlide component and selecting **Design** or **Properties** from its context menu.

Before you begin you must create a **LaunchViewer** function to view the slideshow in Flash Player, if available, or Internet Explorer. Add the following code in the source file to create the function:

- Visual Basic

```

Private Sub LaunchViewer(ByVal filename As String)
 Try
 System.Diagnostics.Process.Start(filename)
 Catch
 System.Diagnostics.Process.Start("IEXPLORE.EXE", filename)
 End Try
End Sub

```
- C#

```

private void LaunchViewer(string filename)
{
 try
 {
 System.Diagnostics.Process.Start(filename);
 }
 catch (Exception e)
 {
 System.Diagnostics.Process.Start("IEXPLORE.EXE", filename);
 }
}

```

### Creating Slide Documents with Navigation Buttons

This topic demonstrates the methods of creating a slideshow that has navigation buttons. To create a four-page slideshow with navigation buttons, complete the following steps:

1. Modify the slide design from the **C1FlashSlide Designer**. For more information on slide layout, see [Using the C1FlashSlide Designer](#) (page 38).
2. To add new pages and draw content to each page, call the AddPage and methods on FPage. Use the following code to create a sample opening page:

- Visual Basic

```
Imports Cl.C1Flash

Private Sub AddPage0()
 ' Create page 0
 Dim page As FPage = C1FlashSlide1.AddPage()

 Dim rect As New Rectangle(100, 100, 350, 80)
 Dim text As String = "This sample demonstrates how to create a
 slideshow using C1FlashSlide component."
 Dim font As New Font("MS Sans Serif", 14)
 page.DrawString(text, font, Brushes.Black, rect)

 rect.Offset(0, 80)
 text = "Some of the pages are copied from other C1FlashCanvas
 samples."
 page.DrawString(text, font, Brushes.Black, rect)

 rect.Offset(0, 80)
 text = "Press <Next> button to go next page."
 page.DrawString(text, font, Brushes.Black, rect)
 font.Dispose()
End Sub
```

- C#

```
using Cl.C1Flash;

private void AddPage0()
{
 // Create page 0
 FPage page = c1FlashSlide1.AddPage();

 Rectangle rect = new Rectangle(100, 100, 350, 80);
 string text = "This sample demonstrates how to create a slideshow
 using C1FlashSlide component.";
 Font font = new Font("MS Sans Serif", 14);
 page.DrawString(text, font, Brushes.Black, rect);

 rect.Offset(0, 80);
 text = "Some of the pages are copied from other C1FlashCanvas
 samples.";
 page.DrawString(text, font, Brushes.Black, rect);

 rect.Offset(0, 80);
 text = "Press <Next> button to go next page.";
 page.DrawString(text, font, Brushes.Black, rect);
 font.Dispose();
}
```

3. Use the following code to create sample pages 1 and 2:

- Visual Basic

```
Private Sub AddPage1()
```

```

' Create page 1
Dim page As FPage = C1FlashSlide1.AddPage()

Dim rect As New Rectangle(100, 50, 350, 80)
Dim text As String = "This page illustrates an image drawn by the
DrawChord method."
Dim font As New Font("MS Sans Serif", 12)
page.DrawString(text, font, Brushes.Black, rect)

Dim a As [Assembly] = [Assembly].GetExecutingAssembly()
Dim an As String = a.GetName().Name
Dim bmp As New Bitmap(a.GetManifestResourceStream((an +
".DrawChordMethodGraphic.bmp")))

page.DrawImage(bmp, New Point(120, 120))
bmp.Dispose()
font.Dispose()
End Sub

```

```
Private Sub AddPage2()
```

```

' Create page 2
Dim page As FPage = C1FlashSlide1.AddPage()

Dim rect As New Rectangle(100, 50, 350, 80)
Dim text As String = "This page illustrates an image drawn by the
DrawEllipse method."
Dim font As New Font("MS Sans Serif", 12)
page.DrawString(text, font, Brushes.Black, rect)

Dim a As [Assembly] = [Assembly].GetExecutingAssembly()
Dim an As String = a.GetName().Name
Dim bmp As New Bitmap(a.GetManifestResourceStream((an +
".DrawEllipseMethodgraphic.bmp")))

page.DrawImage(bmp, New Point(100, 120))
bmp.Dispose()
font.Dispose()
End Sub

```

- **C#**

```

private void AddPage1()
{
 // Create page 1
 FPage page = c1FlashSlide1.AddPage();

 Rectangle rect = new Rectangle(100, 50, 350, 80);
 string text = "This page illustrates an image drawn by the DrawChord
method.";
 Font font = new Font("MS Sans Serif", 12);
 page.DrawString(text, font, Brushes.Black, rect);

 Assembly a = Assembly.GetExecutingAssembly();
 string an = a.GetName().Name;
 Bitmap bmp = new Bitmap(a.GetManifestResourceStream(an +
".DrawChordMethodGraphic.bmp"));

```

```

 page.DrawImage bmp, new Point(120, 120));
 bmp.Dispose();
 font.Dispose();
 }
 private void AddPage2()
 {
 // Create page 2
 FPage page = c1FlashSlide1.AddPage();

 Rectangle rect = new Rectangle(100, 50, 350, 80);
 string text = "This page illustrates an image drawn by the
DrawEllipse method.";
 Font font = new Font("MS Sans Serif", 12);
 page.DrawString(text, font, Brushes.Black, rect);

 Assembly a = Assembly.GetExecutingAssembly();
 string an = a.GetName().Name;
 Bitmap bmp = new Bitmap(a.GetManifestResourceStream(an +
".DrawEllipseMethodgraphic.bmp"));

 page.DrawImage bmp, new Point(100, 120));
 bmp.Dispose();
 font.Dispose();
 }
}

```

**Note:** The bitmap has to be embedded in the Manifest of your assembly. From the **Project | Add New Item** menu, select **Bitmap File** and specify the .bmp file name. Right-click the file located in the Solution Explorer, and select **Properties**. Set the **BuildAction** property to **Embedded Resource**.

4. Sample page 3 uses the drawing routines that appear in the transform section. Use the following code to create sample page 3:

- Visual Basic

```

Private Sub AddPage3()

 ' Create page 3
 Dim page As FPage = C1FlashSlide1.AddPage()

 page.ResetTransform()
 Dim ptCenter As New Point(page.Width / 2, page.Height / 6)

 ' Moves the coordination origin point to the center of the canvas.
 page.TranslateTransform(ptCenter.X, ptCenter.Y)
 Dim rect As New Rectangle(80, 140, 140, 180)

 Dim i As Integer
 For i = 0 To 11

 ' Draws the ellipse with the same ellipse parameter
 page.DrawEllipse(Pens.Red, rect)
 page.FillEllipse(Brushes.LightYellow, rect)
 ' Rotates the coordination by 15 degrees
 page.RotateTransform(15)
 ' Scales the coordination
 page.ScaleTransform(0.85F, 0.85F)
 Next i
}

```

```

' Resets the coordinate
page.ResetTransform()

Dim i As Integer
For i = 1 To 11
 page.DrawEllipse(Pens.Red, rect)

 ' Change the Transform
 Dim m As Matrix = page.Transform
 m.Shear(0.15F, 0.15F)
 page.Transform = m
Next i
End Sub

```

- **C#**

```

private void AddPage3()
{
 // Create page 3
 FPage page = c1FlashSlide1.AddPage();

 page.ResetTransform();
 Point ptCenter = new Point(page.Width/2, page.Height/6);

 // Moves the coordination origin point to the center of the canvas.
 page.TranslateTransform(ptCenter.X, ptCenter.Y);
 Rectangle rect = new Rectangle(80, 140, 140, 180);

 for(int i = 0; i < 12; i++)
 {
 // Draws the ellipse with the same ellipse parameter
 page.DrawEllipse(Pens.Red, rect);
 page.FillEllipse(Brushes.LightYellow, rect);

 // Rotates the coordination by 15 degrees
 page.RotateTransform(15);

 // Scales the coordination
 page.ScaleTransform(0.85F, 0.85F);

 }
 // Resets the coordinate
 page.ResetTransform();

 for(int i = 1; i < 12; i++)
 {
 page.DrawEllipse(Pens.Red, rect);

 // Change the Transform
 Matrix m = page.Transform;
 m.Shear(0.15F, 0.15F);
 page.Transform = m;
 }
}

```

5. To add the pages to create the slideshow, use the AddPage method. Enter the following code in the **Form\_Load** event handler:

- Visual Basic
 

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
 C1FlashSlide1.Clean()

 Me.AddPage0()
 Me.AddPage1()
 Me.AddPage2()
 Me.AddPage3()
End Sub
```
- C#
 

```
private void Form1_Load(object sender, System.EventArgs e)
{
 c1FlashSlide1.Clean();

 this.AddPage0();
 this.AddPage1();
 this.AddPage2();
 this.AddPage3();
}
```

6. Save the canvas to a SWF file and launch it in Internet Explorer. Then save and run your application.

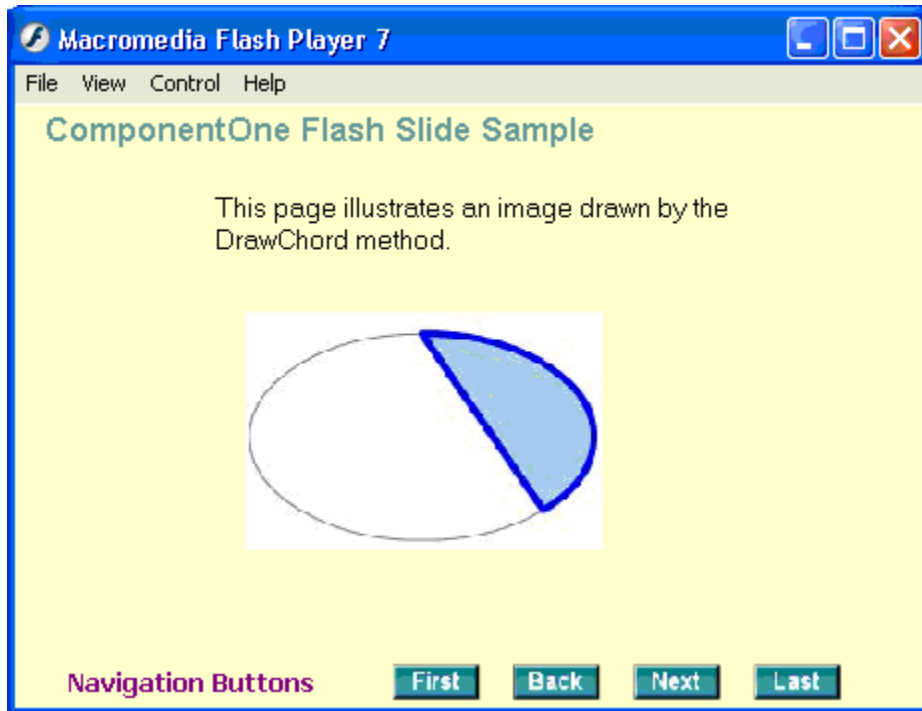
**This topic illustrates the following:**

Here is what your slideshow will look like, page by page.

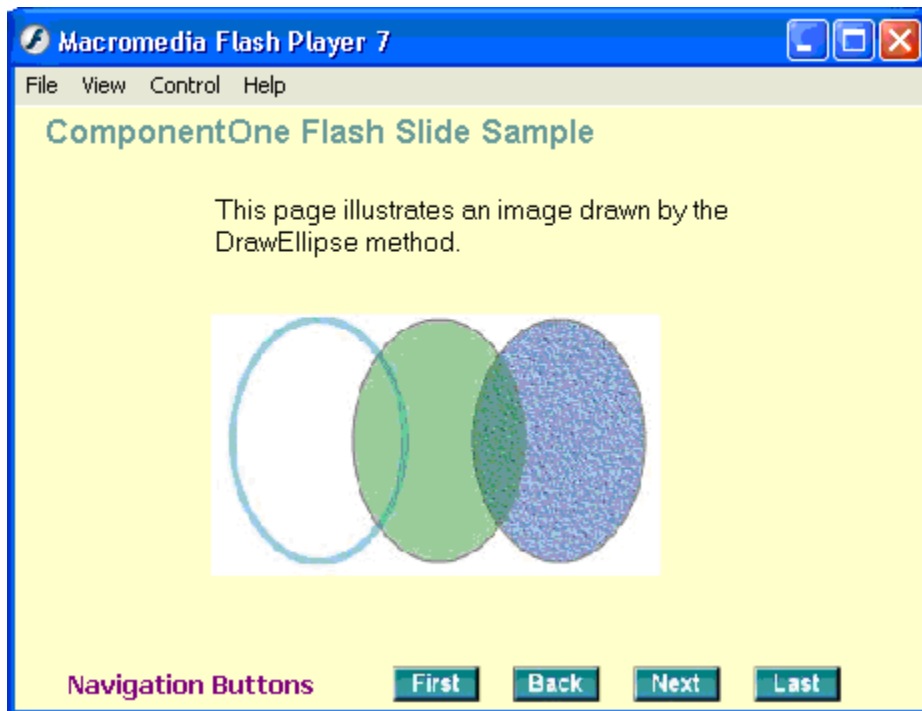
**Opening page**



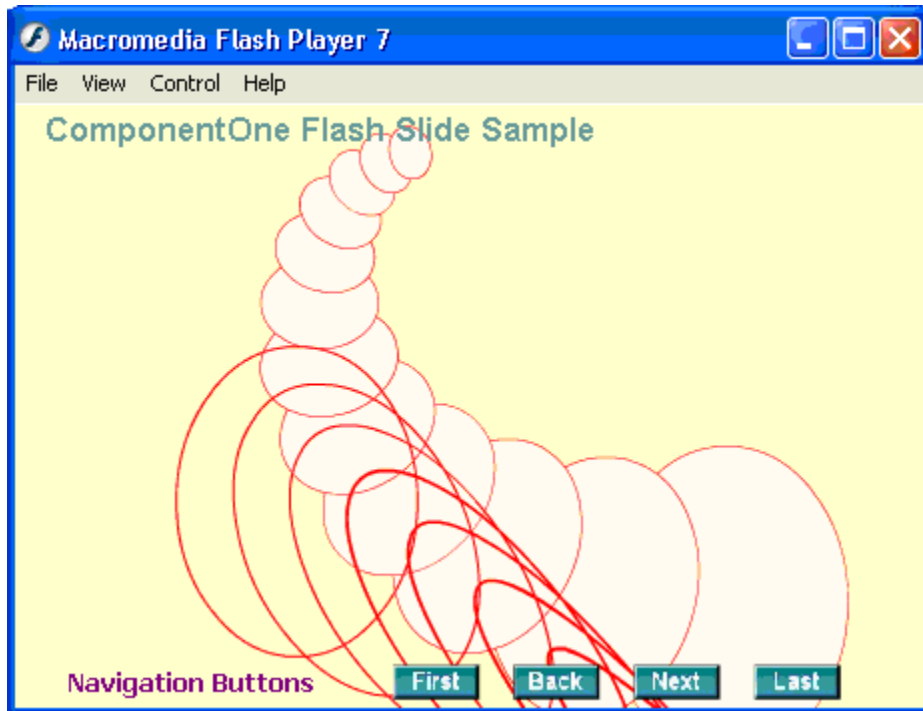
**Page 1**



Page 2



Page 3



## C1WebFlash Class Overview

**C1WebFlash** class allows you to play a Flash movie in variant manners on a browser. This section explains how the **C1WebFlash** control works and shows you how to set up a Web form using the **C1WebFlash** control.

### Using the C1WebFlash Control

**C1WebFlash** is a Web control that controls how a Flash movie is played on a Web page. When rendering the Web control, necessary `<OBJECT>` and `<EMBED>` tags will be inserted into the page. The actual Adobe Flash movie file to be opened and played is referenced in these tags.

The `OBJECT` tag is used by Internet Explorer on Windows and the `EMBED` is used by Netscape Navigator (Macintosh and Windows) to direct the browser to load the Adobe Flash Player. Internet Explorer on Windows uses an ActiveX control to play Adobe Flash content while all other browser and platform combinations use the Netscape plug-in technology to play Adobe Flash content.

Meanwhile, client detection script may be inserted by this control for detecting the existing version of the Flash plug-in.

**C1WebFlash** can also accept any of the ComponentOne's Flash engine components (**C1FlashCanvas**, **C1FlashMovie**, and **C1FlashSlide**) as its content source. When you drop **C1WebFlash** and **C1Flash** engine on to the ASP.NET Web form, clicking the **FlashSource** property in its Property window may drop down a list of Flash engine objects that exist on the form, by which you can choose and "bind" an engine to this Web control.

When the Web control is rendered, the Flash content defined in the Flash engine component will be saved into a temporary folder located under the Web application's virtual path. The correct unique Flash file URL will be inserted into the HTML tag automatically. The temporary files will be cleaned up according to the time span specified by the **SlideExpiration** property.

If you already have a static Flash movie file, you can use the **MovieName** property referring to that file. If both the **MovieName** and **FlashSource** properties are set, **FlashSource** will be used in advance.

# Getting Started with C1WebFlash

The following topics show how to get started using the **C1WebFlash** control.

## Creating an ASP.NET 2.0 Project

When creating ASP.NET 2.0 projects, Visual Studio 2005 gives you the option of creating a Web site project or a Web application project; the latter is similar to creating a Web project in Visual Studio 2003. The Web application project option was provided to help developers converting Web projects from Visual Studio 2003 to Visual Studio 2005.

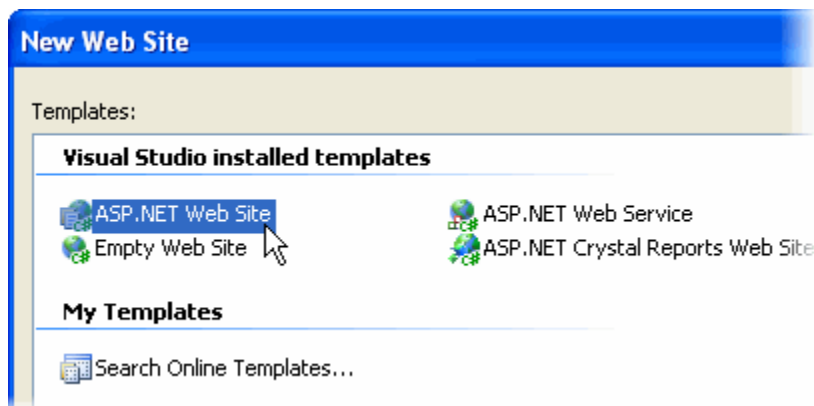
Creating a Web application project requires installation of a Visual Studio 2005 update and add-in, which can be found at <http://msdn.microsoft.com/>. See [Microsoft's Web site](#) for more detailed information and comparisons on Web site and Web application projects.

The steps for creating both types of projects have been provided for your convenience in the Creating a Web Site Project and Creating a Web Application Project topics.

## Creating a Web Site Project

To create a Web Site project, complete the following steps:

1. From the **File** menu in Microsoft Visual Studio .NET, select **New Web Site**. The **New Web Site** dialog box opens.
2. Select **ASP.NET Web Site** from the list of **Templates**.



3. Enter a URL for your application in the **Location** field and click **OK**.

**Note:** The Web server must have IIS version 5 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 Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called `WebForm1.aspx` is displayed in the Web Forms Designer in Design view.

4. Double-click the **C1WebFlash** component from the Toolbox to add it to `Form1`. For information on adding a component to the Toolbox, see [Adding the C1WebFlash Component to a Project](#) (page 72).

Here is the **C1WebFlash** component on the FlashBanner Web form:



## Creating a Web Application Project

To create a new ASP.NET 2.0 Web application project, complete the following steps.

1. From the **File** menu in Microsoft Visual Studio 2005, select **New Project**. The **New Project** dialog box opens.
2. Under **Project Types**, choose either **Visual Basic** or **Visual C#**. Note that one of these options may be located under **Other Languages**.
3. Select **ASP.NET Web Application** from the list of **Templates** in the right pane.
4. Enter a URL for your application in the **Location** field and click **OK**.

**Note:** The Web server must have IIS version 5 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 Forms project is created at the root of the Web server you specified. In addition, a new Web Forms page called `Default.aspx` is displayed in the Web Forms Designer in **Design** view.

5. Double-click the **C1Flash** component in the Toolbox to add it to `WebForm1.aspx`. For information on adding a component to the Toolbox, see *Adding the Flash for .NET Components to a Project*.

## Adding the C1WebFlash Component to a Project

When you install ComponentOne Studio for .NET 2.0, the **Create a ComponentOne Visual Studio Toolbox Tab** checkbox is checked, by default, in the installation wizard. When you open Visual Studio 2005, you will notice a **ComponentOne Studio for ASP.NET 2.0** tab containing the ComponentOne controls has automatically been added to the Toolbox.

If you decide to uncheck the **Create a ComponentOne Visual Studio Toolbox Tab** checkbox during installation, you can manually add ComponentOne controls to the Toolbox at a later time.

**ComponentOne WebFlash for ASP.NET** provides the following control:

- C1WebFlash

To use **C1WebFlash**, add the control to the form or add a reference to the `C1.Web.C1Flash.2` assembly to your project.

### Manually Adding C1WebFlash to the Toolbox

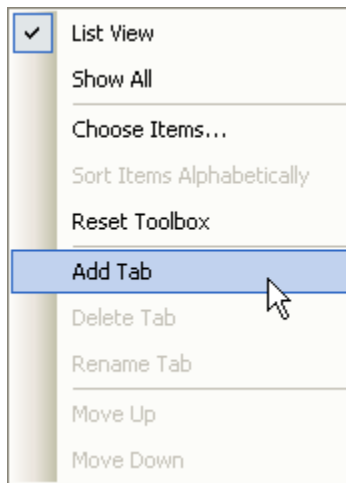
When you install **C1Flash**, the following **C1WebFlash** component will appear in the Visual Studio Toolbox customization dialog box:

- C1WebFlash

To manually add the **C1WebFlash** control to the Visual Studio Toolbox:

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu, if necessary) and right-click it to open the context menu.

2. To make the **C1WebFlash for ASP.NET** component appear on its own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **C1WebFlash**, for example.



3. Right-click the tab where the components are to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
4. In the dialog box, go to the **.NET Framework Components** tab. Sort the list by Namespace (click the Namespace column header) and check the check box for the component belonging to namespace **C1.Web.C1Flash**.

At this point, you should also decide whether you want **C1WebFlash** to run from the GAC (Global Assembly Cache) or locally. If the installation program (or you) places a copy of the component in the GAC, that copy will be used by all applications that use the component. If the component is not installed in the GAC, Visual Studio will make a local copy of the required dlls into your application's bin directory.

Using the GAC can save some disk space, but it also complicates deployment, because you will have to remember to install the components in the GAC on the server as well. In most cases, it is better to remove the component from the GAC and later use XCOPY deployment (the application folder will contain all the .dlls needed to run it). To remove the component from the GAC, open the **WINDOWS\assembly** folder and delete **C1.Web.C1Flash.2** and **C1.C1Flash.2** from it. For more details on the GAC, see the **.NET** documentation.

### Adding C1WebFlash to the Form

To add **C1WebFlash** to a form:

1. Add the **C1WebFlash** control to the Visual Studio Toolbox.
2. Double-click the control or drag it onto your form.

### Adding a Reference to the Assembly

To add a reference to the **C1WebFlash** assembly:

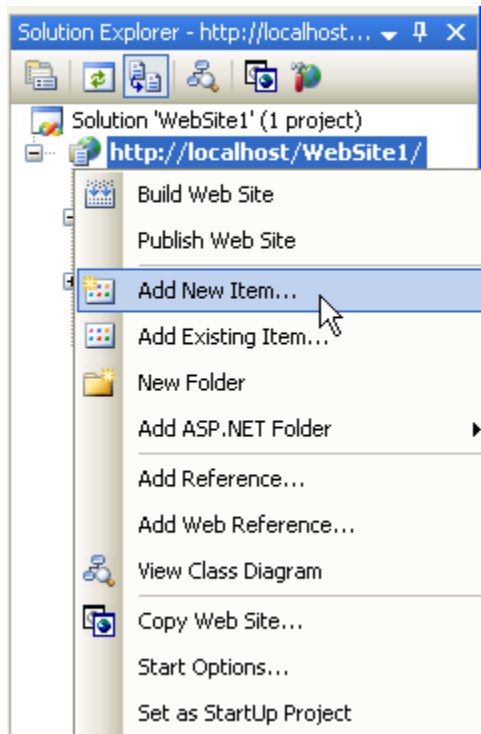
1. Select the **Add Reference** option from the **Website** menu of your Web Site project or from the **Project** menu of your Web Application project.
2. Select the **ComponentOne C1WebFlash** assembly from the list on the **.NET** tab or browse to find the **C1.Web.C1Flash.2.dll** file and click **OK**.
3. Double-click the form caption area to open the code window. At the top of the file, add the following **Imports** statements (**using** in **C#**):  
`Imports C1.Web.C1Flash`

**Note:** This makes the objects defined in the **C1WebFlash** assembly visible to the project. See [Namespaces](#) (page 12) for more information.

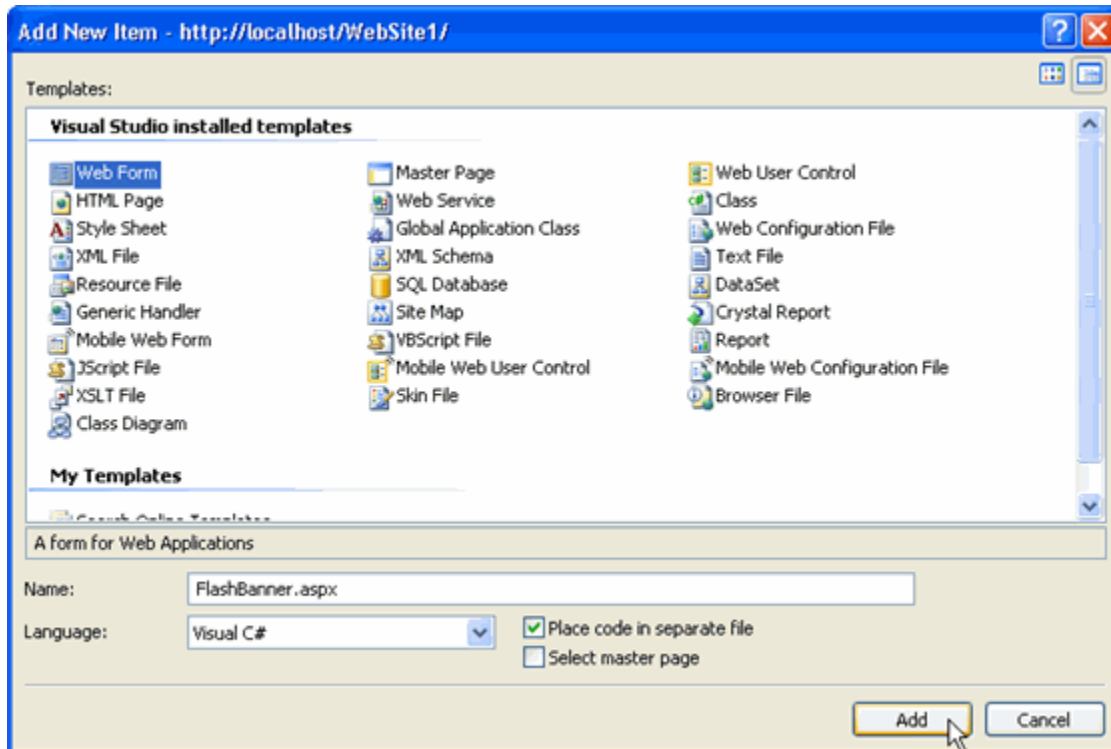
## Creating a New Web Form

To create a new ASP.NET form, complete the following steps:

1. In the Solution Explorer, right-click the Web site project and select **Add New Item** from its context menu.



2. The **Add New Item** dialog box appears. Select **Web Form** from the list of templates and type in the form's name, **FlashBanner.aspx**, for example.

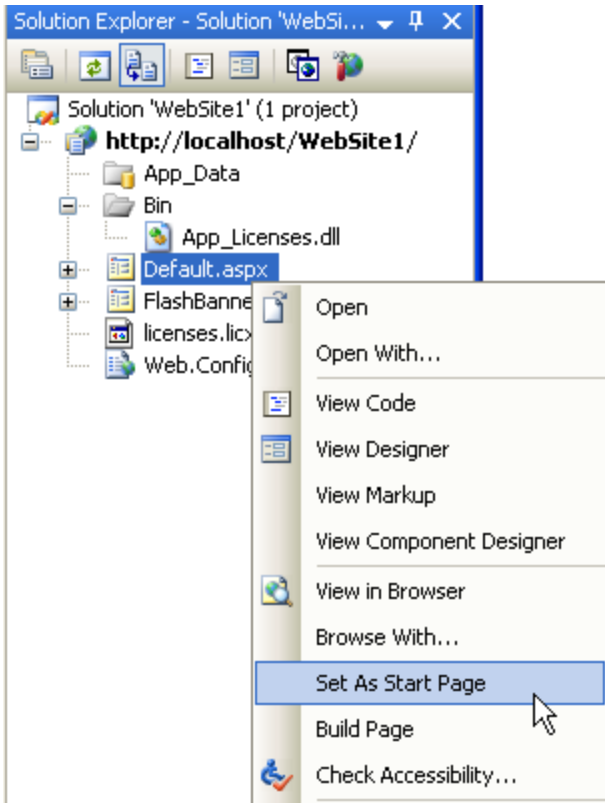


3. Click **Add**. The **FlashBanner.aspx** page is added to your project.

## Setting the Start Page for Your Web Application

Before running the Web project, you have to select the Start page.

1. Right-click the Web form to use as a start page, **Default.aspx**, for example.

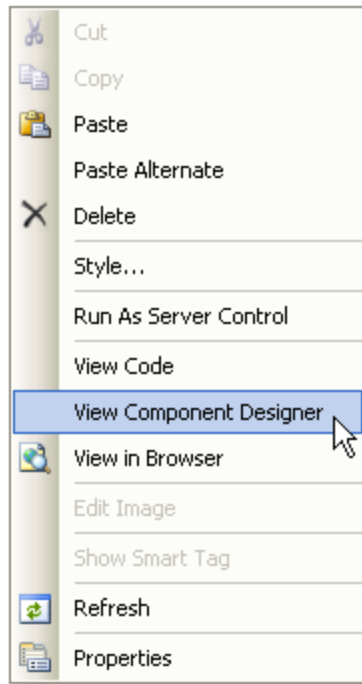
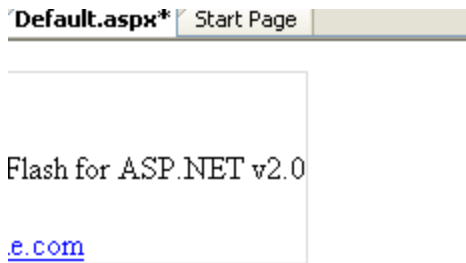


2. Run the project and observe that the **Default.aspx** page opens in the browser.

### Adding the C1Flash Components to Your Web Application

To add the **ComponentOne Flash for .NET** components to your Web form, you must open the Design page. To open the Web form's Design page:

1. Right-click your .aspx form and select **View Component Designer** from its context menu.



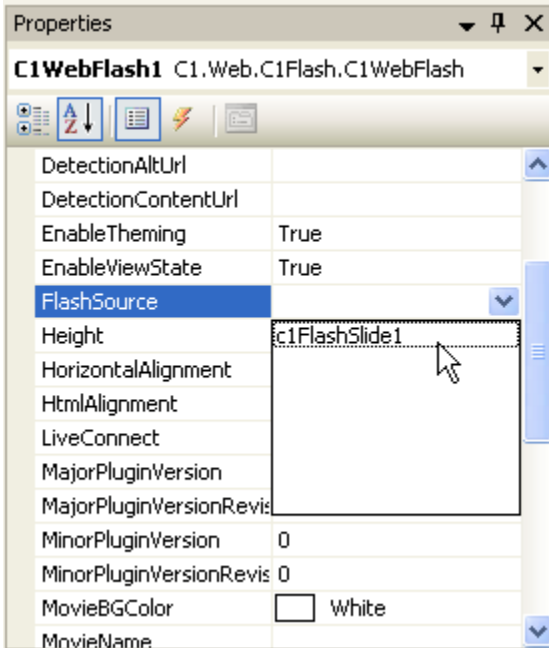
Now your WinForms components appear in the Toolbox. If you have not added the **C1Flash** components to the Toolbox yet, please see [Adding the Flash for .NET Components to a Project](#) (page 14).

2. Add the **C1Flash** component you would like to use to your form.
3. Click the `.aspx` tab to exit the Component Designer.

### Binding a Flash for .NET Component to the C1WebFlash Control

After you have added a **ComponentOne Flash for .NET** component to your Web form, you must connect the Flash engine component to the Web control. To bind the component to the **C1WebFlash** control, complete the following steps:

1. Select the **C1WebFlash** component on your form and locate the `FlashSource` property in the Property grid.
2. Set the `FlashSource` property to the **C1FlashCanvas**, **C1FlashSlide**, or **C1FlashMovie** component depending on what Flash engine component you would like to reference.



Note that only **C1FlashSlide** appears in the **FlashSource** drop-down list since it was the only component that was added in the Component Designer.

## C1WebFlash Samples

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

### C# Samples

| Sample   | Description                                                                                       |
|----------|---------------------------------------------------------------------------------------------------|
| AdBanner | Demonstrates how to dynamically generate an advertisement banner in Flash format onto a Web page. |
| Chart    | Demonstrates how to output a Chart graph in Flash format onto a Web page.                         |