CA Gen

ASP.NET User Guide

Release 8.5

CA Technologies Product References

This document references the following CA Technologies products:

- CA Gen
- AllFusion® Gen

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Overview

The ASP.NET Web Client feature of CA Gen uses the Web as an access medium to connect to CA Gen servers. ASP.NET Web Clients build on Microsoft .NET technologies. The application logic is generated in C# and the user interface downloaded to browsers is HTML with JavaScript. Users who require Web browser access to CA Gen servers will benefit from reading this guide. This guide describes design, generation, and building by providing the following information:

- Prerequisites for installing the ASP.NET Web Client software
- Technical requirements for ASP.NET Web Clients
- How to install the new software
- How to generate the application
- How to build files and use the application
- Consistency check messages

.NET Overview

Microsoft® .NET is a set of Microsoft software technologies for connecting information, people, systems, and devices. It spans clients, servers, and developer tools.

The following components are primarily used by CA Gen ASP.NET Web Clients:

- Microsoft Visual Studio® .NET, which provides a rapid application integrated development environment for programming with the .NET Framework
- .NET Framework

.NET Framework

The .NET Framework is the programming model of the .NET environment for building, deploying, and running Web-based applications, smart client applications, and XML Web services. The key components of the .NET Framework are the common language runtime and the .NET Framework class library, which includes ADO.NET, ASP.NET, and Windows Forms.

Common Language Runtime

The Common Language Runtime (CLR) is responsible for runtime services such as language integration, security enforcement, memory management, process management, and thread management. In addition, it has a role at development time when features such as life-cycle management, strong type naming, cross-language exception handling, and dynamic binding reduce the amount of code that a developer must write to turn business logic into a reusable component.

Framework Class Libraries

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object-oriented, providing types from which your own managed code can derive functionality. Base classes provide standard functionality such as input/output, string manipulation, security management, network communication, thread management, text management, user interface design features, and other functions. The Microsoft ADO.NET data classes support persistent data management and include SQL classes for manipulating persistent data stores through a standard SQL interface. XML classes enable XML data manipulation and XML searching and translations. The Microsoft ASP.NET classes support the development of Web-based applications and XML Web services. The Windows Forms classes support the development of Windows-based smart client applications. Together, the class libraries provide a common, consistent development interface across all languages supported by the .NET Framework.

Common Type System

The common type system defines how types are declared, used, and managed in the runtime, and is also an important part of the runtime's support for cross-language integration. The common type system performs the following functions:

- Establishes a framework that enables cross-language integration, type safety, and high performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.
- Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.

Common (Microsoft) Intermediate Language

One of the biggest advantages that the runtime framework provides is a language-neutral execution environment. All code, irrespective of the source language, is compiled automatically into a standard intermediate language (IL) - either on command or when first executed (in the case of ASP.NET). The runtime framework then creates the final binary code that makes up the application and executes it. The compiled IL code is used for each request until the source code is changed, at which point the cached version is invalidated and discarded.

Common Language Specification

To fully interact with other objects regardless of the language they were implemented in, objects must expose to callers only those features that are common to all the languages they must interoperate with. For this reason, a set of language features called the Common Language Specification has been defined.

Assembly

An assembly is a collection of one or more files grouped together to form a logical unit. Most assemblies contain just one file, but assemblies can and sometimes do include multiple files. All the files that make up a multifile assembly must reside in the same directory. When you use the C# compiler to produce a simple EXE, it is referred to as an assembly.

Global Assembly Cache (GAC)

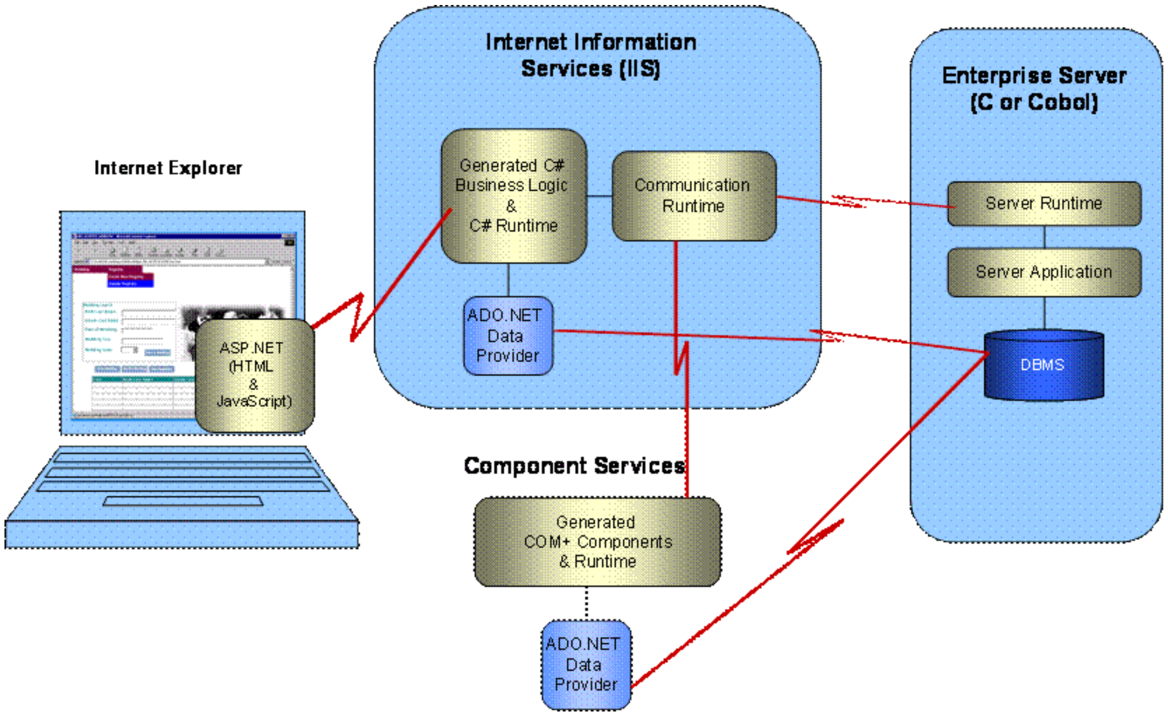
To share an assembly where any application can find it, it must be installed in the Global Assembly Cache (GAC). The GAC is a repository for shared assemblies. When the CLR attempts to load an assembly, it looks in the GAC even before it looks in the local application directory.

ASP.NET Web Clients

ASP.NET Web Clients are applications generated by CA Gen. They use Microsoft ASP.NET technology, which provides a unified Web development model that includes the services necessary to build enterprise-class Web applications. ASP.NET is part of the .NET Framework and can take full advantage of the features of the common language runtime, such as type safety, inheritance, language interoperability, and versioning.

ASP.NET Web Client Architecture

ASP.NET Web Clients generate thin client applications, that are based on the GUI design model. The resulting client application logic resides primarily on the Web server, Internet Information Services (IIS), while the interface is downloaded to an Internet Explorer web browser. The following illustration shows the high level architecture of the CA Gen-generated ASP.NET Web Client application:


Internet Information Services (IIS)
Enterprise Server (C or Cobol)
Internet Explorer
Generated C# Business Logic & C# Runtime
Communication Runtime
Server Runtime
Server Application
ADO.NET Data Provider
DBMS
ASP.NET (HTML & JavaScript)
Component Services
Generated COM+ Components & Runtime
ADO.NET Data Provider

ASP.NET Web Client Environments

The following five environments are involved in ASP.NET Web Client generation, and these environments can be located on one machine or five physically separate machines:

- Server environment

  The CA Gen runtime is installed for generated server applications. No special setup is required for ASP.NET Web Client applications.

- Application Server environment

  The runtime engine and application logic for the client is installed on an Internet Information Services (IIS) Application Server, which enables CA Gen to run the generated ASP.NET Web Client application.

- Development environment

  The CA Gen Toolset provides the environment to develop and generate CA Gen ASP.NET Web Client applications. This environment contains the components that generate the application both locally and remotely. The client Server Encyclopedia can also be used to generate ASP.NET applications.

- Build environment

  This environment enables the generated files from the CA Gen Development environment to be compiled and packaged into an MSI file. The CA Gen Build environment is often located on the same machine as the CA Gen Development environment.

- End user environment

  This environment enables end users to access CA Gen ASP.NET Web client applications through an Internet Explorer browser.

ASP.NET Web Client Highlights

The following list discusses the highlights of ASP.NET Web Client applications that are generated using CA Gen:

- The application logic resides on the application server machine.
- ASP.NET Http Handlers (ASHXs) serve as entry points to generated applications.
- The User Interface is a combination of HTML, JavaScript, and Cascading Style Sheets that are created dynamically by the Gen runtime and downloaded to the browser.
- Permitted Value and Edit Pattern checking takes place on the browser using JavaScript.
- The product lets existing users reuse their models by generating ASP.NET Web Client applications.
- ASP.NET Web Client software can access databases by using Gen Server applications or using ADO.NET on the Application Server environment.
- ASP.NET Web Client user interface must be on a Windows platform using a supported version of Internet Explorer.
- TCP/IP, MQSeries, Web Services, and .NET Remoting are the only thread-safe middleware available for ASP.NET Web Client.

Note: Due to browser and HTML limitations, some CA Gen-related features have a different level of support than GUI applications when generated for ASP.NET Web Clients.

More Information:

Unsupported Features (see page 65)

Related Information

For more detailed information on ASP.NET applications or related topics, see the following:

Prosise, Jeff. *Programming Microsoft .NET*. Microsoft Press, 2002.

For more information on ASP.NET web applications, see the relevant Microsoft documentation.

Note: Throughout this document, the environment variable %GEN*xx*%Gen is used. *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

Chapter 2: Pre-Installation Planning

This chapter provides the infrastructure requirements you need to know as you prepare to install the ASP.NET Web Client software. The four different environments associated with this feature can be located on one machine or four physically separate machines. The end-user machine must have the supported version of Internet Explorer.

Note: For more information about the supported versions of Internet Explorer, see *Technical Requirements* available at http://www.ca.com/support.

Technical Requirements

Third-Party software and CA Gen software each have requirements in addition to the ASP.NET Web Client requirements shown in the following table:

Environment	Third-Party Software	CA Gen Software
CA Gen Server	No change required for ASP.NET Web Client applications.	Installed CA Gen cooperative server.
Application Server	Internet Information Services (IIS) .NET Framework	ASP.NET Web Client*. Communications Runtime**
Development	Windows Operating System ** Internet Information Services (IIS) **, ***	Workstation Development Toolset or Client/Server Encyclopedia (CSE) for generation only. ASP.NET Web Client
Build	Windows Operating System ** Visual Studio**	Implementation Toolset ASP.NET Web Client
End User	Microsoft Internet Explorer**	

Note:

- * Only needed if not already installed in the GAC.
- ** See the *Technical Requirements* available on http://ca.com/support for details.
- *** Internet Information Services (IIS) is only needed if third-party Web Controls are used in a model.

Supported Features

Due to limitations of the browser and HTML, some features have a different level of support than GUI applications when generated using CA Gen for ASP.NET Web Client.

More Information:

Unsupported Features (see page 65)

Chapter 3: Installing ASP.NET Web Client

This chapter reviews the installation procedure for the ASP.NET Web Client feature. Before you start the installation, see Technical Requirements (see page 15).

Installation Procedure for ASP.NET Web Client

You must install ASP.NET Web Client as instructed from the CA Gen download folder to four different environments. These environments can be located on one machine or four physically separate machines.

Set Up the Development Environment

You must set up the development environment to generate and install ASP.NET Web Client applications.

Follow these steps:

1. Set up your environment to generate and install the ASP.NET Web Client component, in addition to other CA Gen development tools such as the Workstation Toolset.
2. Set up any third-party Web Controls used in the application.

Note: Steps 2 is only needed if third-party Web Controls are used in a model.

Set Up the Build Environment

The Build machine is where the application is built and packaged in preparation for installation. ASP.NET Web Clients can only be built on Windows platforms, so only the Implementation Toolset windows can be used.

This Implementation Environment must have the Implementation Toolset installed.

The following components are needed for ASP.NET Web Clients:

- Build Tool

  Builds the source code generated from the Workstation or the Client Server Encyclopedia.

- Assemble Option

  An option available on the Build Tool. It is required to package ASP.NET Web Client applications in preparation for installation on an IIS Application Server. Assembling creates an .MSI file that contains all the required components to install and run ASP.NET Web Client applications.

- Communication Runtime Software
  - .NET Remoting middleware, if flowing to a .NET Server
  - TCP/IP middleware, if flowing to a CA Gen server using TCP/IP
  - MQSeries middleware, if flowing to a CA Gen server using MQSeries
  - Web Services middleware, if flowing to a CA Gen server using Web Services
- Third-party Software
  - Visual Studio
  - Any third-party Web Controls used in the generated application
  - The following if DDL installation is needed:
- Database
- ODBC driver

Set Up the Application Server Environment

ASP.NET applications use encrypted config files as the default assemble option and this option requires IIS to use RSA key container. The access to RSA Key container needs to be explicitly granted to the user account that runs ASP.NET applications. The user account to run ASP.NET application is different on each version of IIS. For example, the default user account for each IIS is listed in the table next:

IIS Version	Default User Account
IIS 7.0, Windows Server 2008	NETWORKSERVICE
IIS 7.5, Windows Server 2008 R2 and Windows 7	NETWORKSERVICE

The user account is configurable on IIS 7 with application pool configuration panel.

The following command is used to grant the access to RSA Key container.

```
aspnet_regiis -pa "NetFrameworkConfigurationKey" "<user account>"
```

You must set up the Application Server environment to generate and install ASP.NET Web Client applications.

Follow these steps:

1. Invoke the setup program created by the Build Tool Assemble Option. The setup program installs the contents of the .MSI file under the Application Server automatically.
2. Install third-party Web Controls, if any are required.

Note: For more information about Microsoft Software setup, see Microsoft documentation.

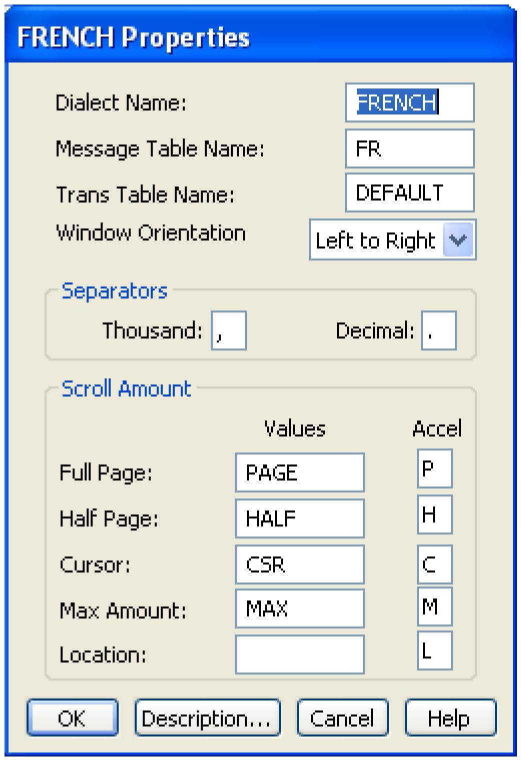
Set Up the Message Resources

After installing the ASP.NET Web Client option, the message resource files automatically install in the <CA Gen install directory>\.net\localization\resources directory. This directory must contain, but may not be limited to the following files:

- csumessages.txt
- fmrtmessages.txt

Localization of these files uses the naming convention <manager type>+ "messages" + "." + <two letter language identifier> + ".txt".

CA Gen provides localized versions for a set of languages. If needed, you can create a variant of the resource file. The language identifier used is the two letters defined in the model under Dialect Definition, Properties, and Message Table Name.



Dialect Name:	FRENCH
Message Table Name:	FR
Trans Table Name:	DEFAULT
Window Orientation	Left to Right

Separators

Thousand:	,
Decimal:	.

Scroll Amount

	Values	Accel
Full Page:	PAGE	P
Half Page:	HALF	H
Cursor:	CSR	C
Max Amount:	MAX	M
Location:		L

Rules to Edit Message Resource Files

The following syntactical rules apply when editing the message resource files:

- Set all parameters using <parameter>=<value> (with no spaces before or after the equals sign).
- Enclose variables in curly braces {} and increment as needed. The increment is zero relative.

- Use # for comments.
- Use \ for line continuations.

Set Up the End-User Environment

ASP.NET Web Client application end users can access the application using Internet Explorer. No special setup is necessary.

Chapter 4: Designing an ASP.NET Web Client Application

After installing the software, you are ready to use the Toolset to define ASP.NET Web Client components.

This section contains the following topics:

ASP.NET Web Client Modeling

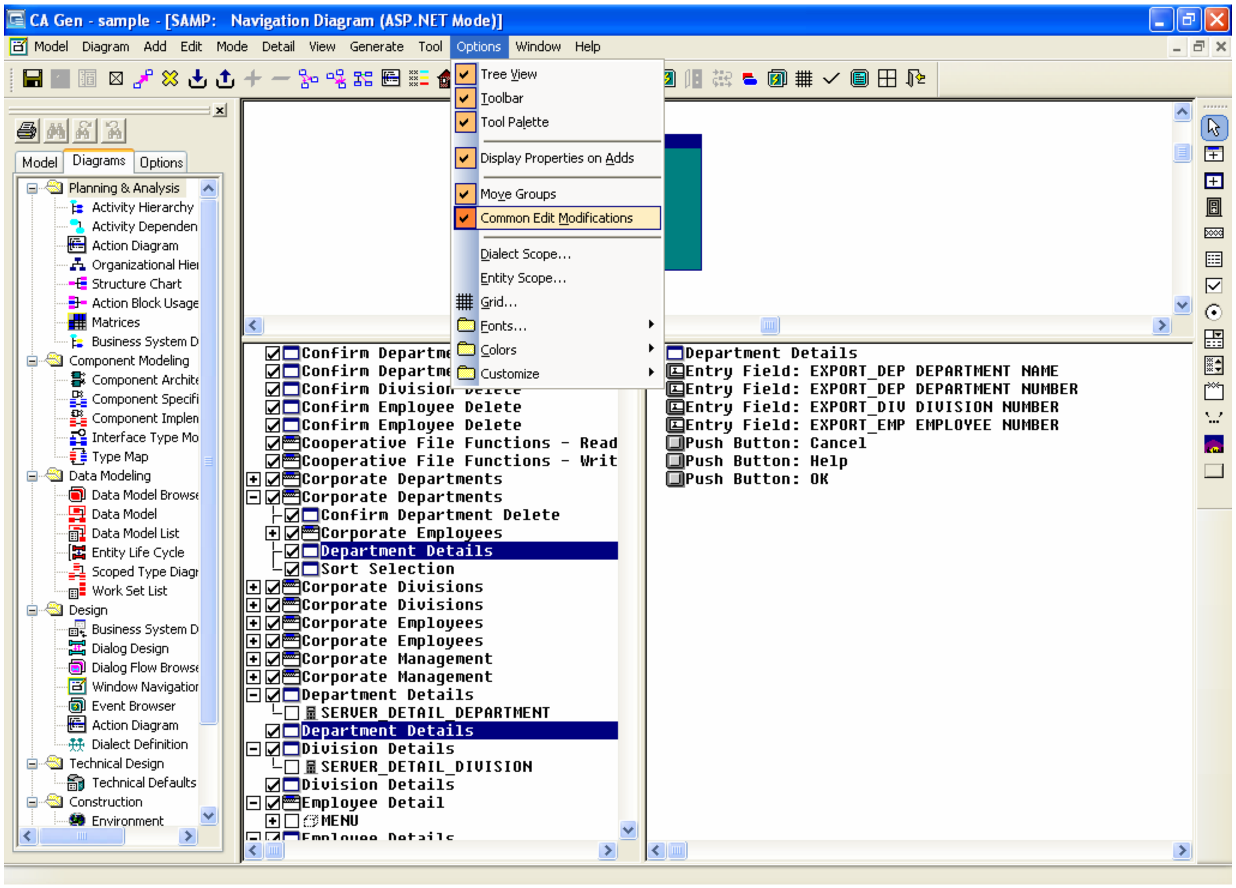
The modeling for ASP.NET Web Client applications closely mimics that of GUI and Java Web Generation applications. The three types of applications share the same Action Diagram logic. GUI design is the basis for the user interface. However, you can also use specialized features for the other two target environments.

Common Edit Modifications

With the Common Edit Modification menu, you can make common edits to a window across all the modes. They are:

- ASP.NET
- HTML
- Window

Moving a control in the Edit Window mode moves the field when the window displays in either Edit HTML mode or Edit ASP.NET mode. The same is true when editing a field in Edit HTML mode or Edit ASP.NET mode. This *common* functionality is the default for the Common Edit Modifications menu item and occurs when this option is checked.

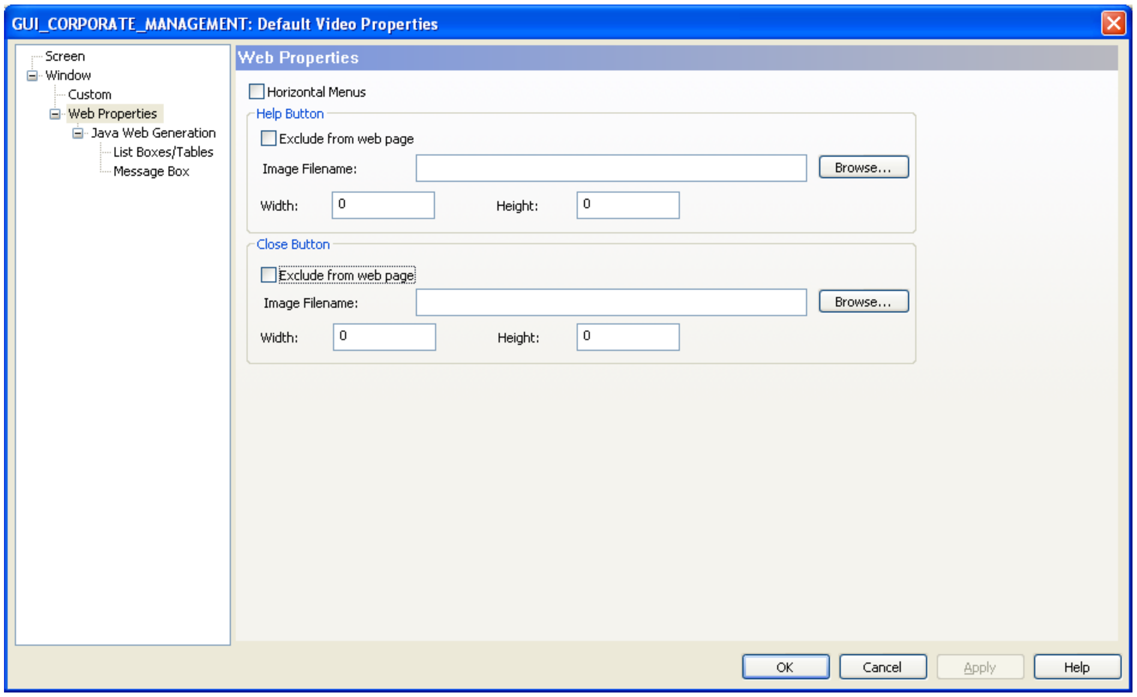


Department Details
Entry Field: EXPORT_DEP DEPARTMENT NAME
Entry Field: EXPORT_DEP DEPARTMENT NUMBER
Entry Field: EXPORT_DIV DIVISION NUMBER
Entry Field: EXPORT_EMP EMPLOYEE NUMBER
Push Button: Cancel
Push Button: Help
Push Button: OK

If Common Edit Modifications is not checked, any edits causing updates to any of the visible properties of a window and its controls are considered a customization for the edit mode used in the current window (Windows, HTML, or ASP.NET). Control location, prompt text, control font, and color are examples of customizable properties. You can also add, update, or delete a control relative to one mode only.

Web Properties

The Web Properties node on the Business System Defaults Video Properties dialog provides configurable options at the level of Business System. You can specify the menu styles, and the images for Help, and Close buttons.



GUI_CORPORATE_MANAGEMENT: Default Video Properties

- Screen
- Window
  - Custom
  - Web Properties
    - Java Web Generation
      - List Boxes/Tables
      - Message Box

Web Properties

☐ Horizontal Menus

Help Button

☐ Exclude from web page

Image Filename:

Width: 0 Height: 0

Close Button

☐ Exclude from web page

Image Filename:

Width: 0 Height: 0

Menus

GUI applications support one style of menus only, horizontal. In ASP.NET Web Client applications, you have the choice of selecting whether to generate vertical menus (starting at the top left corner and expanded vertically down the side of the page) or horizontal menus (starting at the top left corner and expanded horizontally across the page). The default style is vertical menus. The option to switch between both styles is available on the level of the Business System.

Customizable Help and Close Buttons

ASP.Net Web Client Generation can create a Help button with a value of '?' and a Close button with a value of 'X' on a window if its Main Menu Bar's property: Generate Window and Help Menus is selected. Those buttons also get generated for Dialogs when the System Menu property is selected. You can select at the Business System Level whether or not you want to generate these Help and Close buttons. At the Business System Level, you can also optionally place a bitmap of your choice for Help and Close Buttons, which will apply to all windows in the Business System.

You can browse and select the required images for the Help and Close buttons from the Business System Defaults Video Properties dialog. The Toolset will copy them into the html/images directory. The image will not be displayed at design time.

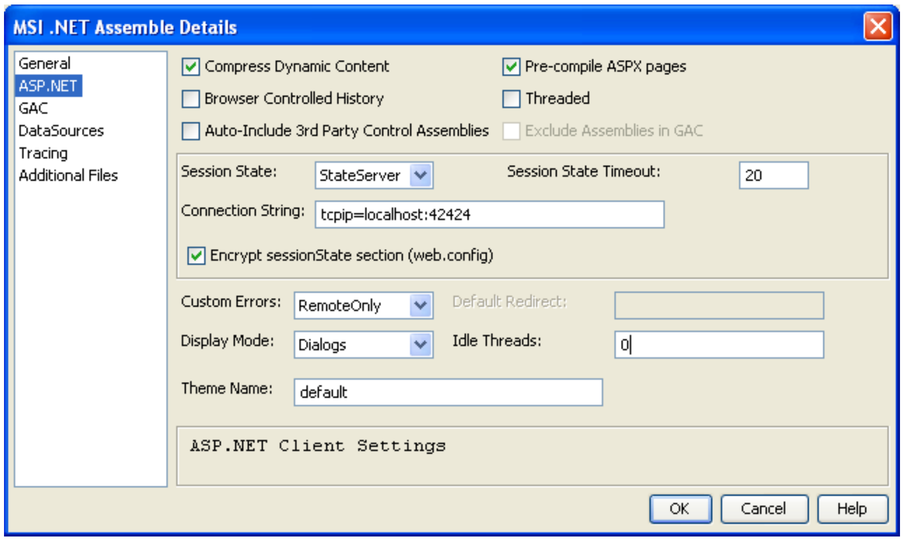
For more information about the Business System Defaults Video Properties dialog, see the *Toolset Help.*

Themes

A theme is a collection of appearance settings that you can define, and consistently apply across the application, and the application control types. You can apply a single theme across all the pages of CA Gen ASP.NET application. A theme for a CA Gen ASP.NET application consists of control skins, CSS files, and image files.

You can customize the appearance of ASP.NET applications by applying changes to the theme, without changing the appearance settings for individual windows, or controls in the application.

With the MSI .NET Assemble Details dialog in the Build Tool, you can specify the name of the theme, <*themename*>, for an ASP.NET application. The following image details the MSI.NET Assemble Details dialog in the Build Tool:



[✓] Compress Dynamic Content
[✓] Pre-compile ASPX pages
[ ] Browser Controlled History
[ ] Threaded
[ ] Auto-Include 3rd Party Control Assemblies
[ ] Exclude Assemblies in GAC

Session State: StateServer
Session State Timeout: 20
Connection String: tcpip=localhost:42424
[✓] Encrypt sessionState section (web.config)

Custom Errors: RemoteOnly
Default Redirect:
Display Mode: Dialogs
Idle Threads: 0
Theme Name: default

ASP.NET Client Settings

Note: For more information about the MSI.NET Assemble Details dialog, see the *Build Tool Help*.

When you specify a theme, CA Gen creates a directory, APP_Themes\<*themename*>, in the application root directory when installing the application on the server. This directory stores the control skin files, the CSS files, and the image files for each theme. User can create, add, and maintain control skins, CSS files, or image files for each theme in this directory.

The web.config file in the application root directory includes a pages tag with the theme attribute, *<pages theme=”<themename>” />*.

Skins

Skins are visual attribute definitions for application controls that are stored in the skin file, *[set the File Name variable].skin*. Control skins define only the visual attributes that are a part of a theme for an application.

In CA Gen ASP.NET applications, you can define control skins for each control type in a separate file, or you can define all the control skins for a theme in a single file. The <*themename*> directory for a theme must include all the control skins that are a part of the theme.

The control skin *overrides* the appearance settings for the control in the Navigation diagram. However, the appearance settings of a control through dynamic attributes or Dot Notation *override* the settings in the control skin.

CA Gen supports default control skins. A default control skin automatically applies to the all the controls of the same type when a theme is applied to the ASP.NET application. A default skin has the following format:

```
<ServerControlType runat=”server” property1=”value1” property2=”value2” ……/>
```

For example, this is the default control skin entry for a single line entry field:

```
<Gen:ASPXSingleLineEdit runat="server" Height="20px" Width="100px" ForeColor="Red"
BackColor="#FFE4C4" Font-Name="Script" Font-Size="14pt" Font-Bold="false"
Font-Italic=”true” Font-Strikeout=”false” Font-Underline=”true”
BorderStyle="groove" BorderColor="#FF0000" BorderWidth="1pt"/>
```

This default skin applies to all SingleLineEdit controls on pages that use the theme.

CA Gen provides a sample skin, Gen_Sample.skin, in the CA Gen samples\ASP.NET directory.

Note: CA Gen does not support named skins that apply to a specific instance of a control type.

Rules to Specify Visual Attributes in Control Skins

When you specify the visual attributes for a control skin, you can specify only the visual attributes that are supported for each control type.

The following rules apply when specifying the visual attributes for control skins that support a theme in an ASP.NET application:

1. The theme for an ASP.NET application can support only the following *common* visual attributes:
   - Width
   - Height
   - ForeColor
   - BackColor
   - Font
     - FontType (Font-Name, or Font-Names)
     - Font-Size
     - Font Style properties

       (Font-Bold, Font-Italic, Font-Strikeout, Font- Underline, Font-Overline).

       The value of Font Style properties can be either *true* or *false.*
   - Border
     - BorderColor
     - BorderStyle (includes dotted, dashed, solid, double, groove, ridge, inset, and outset.)
     - BorderWidth

   Note: The color for the ForeColor, BackColor and BorderColor attributes may be represented by the name of the color or the hexadecimal RGB code for the color prefixed with #.

2. The control skins for the individual control types may support one, more, or all of the *common* visual attributes in the theme.

   The following table lists the supported common visual attributes in a control skin for each control type in a CA Gen ASP.NET application:

Control Type	Supported Common Visual Attributes	Unsupported Common Visual Attributes
asp:Label	All the common visual attributes.	

Control Type	Supported Common Visual Attributes	Unsupported Common Visual Attributes
Gen:ASPXBitmap	Width, Height, Border	ForeColor, BackColor
Gen:ASPXBitmapButton	Width, Height, Border	ForeColor, BackColor
Gen:ASPXButton	All the common visual attributes.	
Gen:ASPXCheckBox	All the common visual attributes.	
Gen:ASPXRadioButton	All the common visual attributes.	
Gen:ASPXSingleLineEdit	All the common visual attributes.	
Gen:ASPXMultiLineEdit	All the common visual attributes.	
Gen:ASPXGroupBoxLiteral	All the common visual attributes.	
Gen:ASPXStaticLiteral	All the common visual attributes.	
Gen:ASPXPermValueList	ForeColor,BackColor,Width, Font	Height, Border
Gen:ASPXHyperLink	All the common visual attributes.	
Gen:ASPXLinkButton	All the common visual attributes.	
Gen:ASPXStatusBar	All the common visual attributes.	
Gen:ASPXToolBar	All the common visual attributes.	
Gen:ASPXDropDownNoEntry	ForeColor,BackColor,Width, Font	Height, Border
Gen:ASPXListBox	ForeColor, BackColor, Font, Border	Width, Height
Gen:ASPXFixedSizeTable	ForeColor, BackColor, Font, Border	Width, Height
Gen:ASPXVaryingSizeTable	ForeColor, BackColor, Font, Border	Width, Height
Gen:ASPXListBoxColumn (listbox/table prompt)	ForeColor, BackColor, Font, Border, Height	Width

Control Type	Supported Common Visual Attributes	Unsupported Common Visual Attributes
Gen:ASPXMenuBar	ForeColor, BackColor, Font, Width	Height, Border

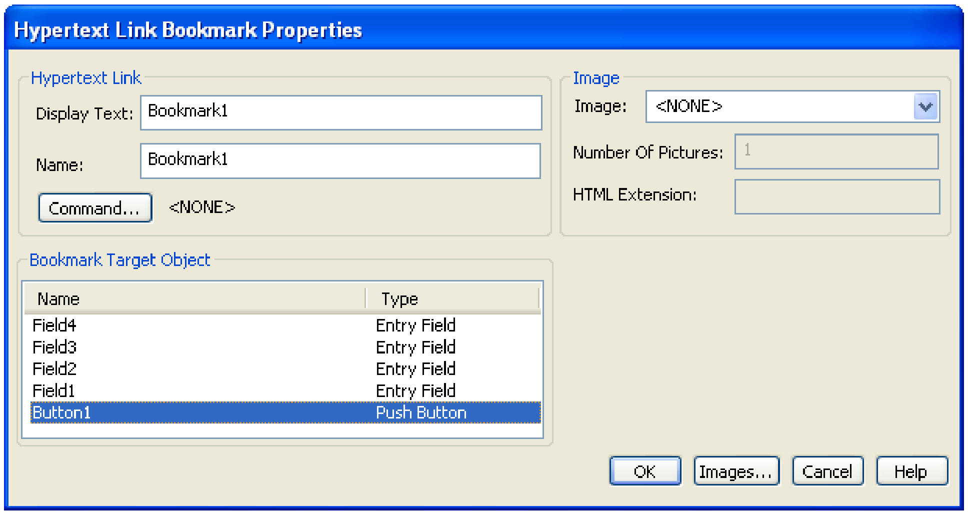
Hypertext Links

Hypertext links will be common to both HTML and ASP.NET modes even if the Common Edit Modifications Option is not selected. ASP.NET Web Clients and Java Web Generation applications support two types of hypertext links:

- Bookmark-This option links to a location on the current page.
- Open New Browser Window-This option opens a new browser window with a user-specified page and transfers control to this new window.

Both types of hypertext links are available in ASP.NET and HTML modes, but are not available in Window mode. They are accessed from the Add menu item in the Navigation Diagram.

To access the Bookmark option, choose Hypertext Link from the Add menu, and click Bookmark. The Hypertext Link Bookmark Properties dialog appears:



Hypertext Link Bookmark Properties

Hypertext Link
Display Text: Bookmark1
Name: Bookmark1
Command... <NONE>

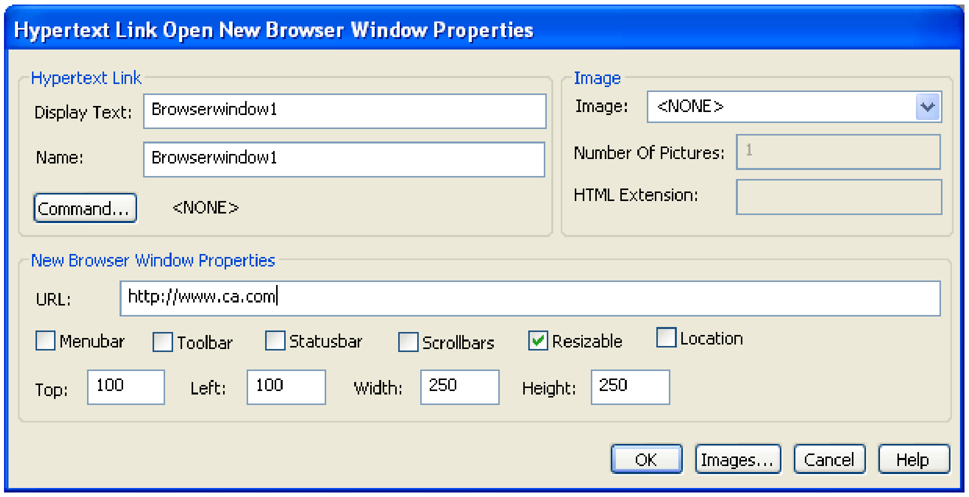
Image
Image: <NONE>
Number Of Pictures: 1
HTML Extension:

Bookmark Target Object

Name	Type
Field4	Entry Field
Field3	Entry Field
Field2	Entry Field
Field1	Entry Field
Button1	Push Button

The Display Text and Name fields are populated with default values that you can replace. Note that the bookmark name must be unique within the scope of its parent window. You must specify the target object the bookmark will link to. The Target Object can be any control on the window.

To access the Open New Browser Window option, choose Hypertext Link from the Add menu, and click Open New Browser Window. The Hypertext Link Open New Browser Window Properties dialog appears.


Hypertext Link Open New Browser Window Properties

Hypertext Link

Display Text: Browserwindow1

Name: Browserwindow1

Command... <NONE>

Image

Image: <NONE>

Number Of Pictures: 1

HTML Extension:

New Browser Window Properties

URL: http://www.ca.com

☐ Menubar ☐ Toolbar ☐ Statusbar ☐ Scrollbars ☑ Resizable ☐ Location

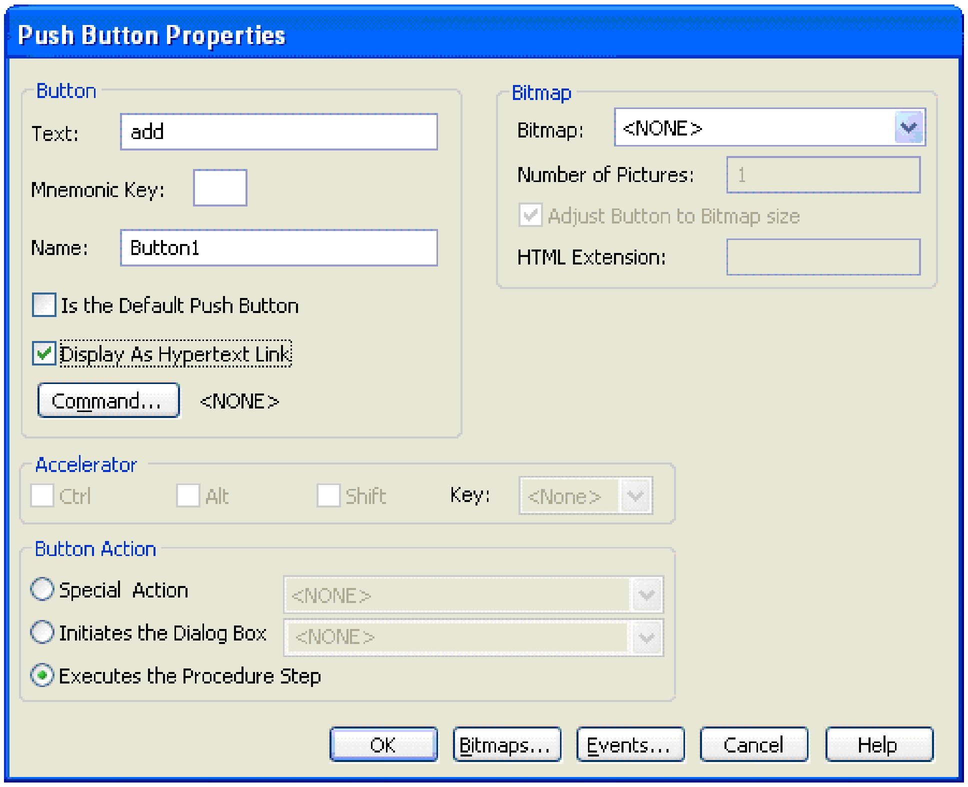
Top: 100 Left: 100 Width: 250 Height: 250

The Name and Display fields are populated with a default you can change. The URL field is required. You can select properties for the window by checking any of the check boxes under the URL field. For example, if you check Location, the browser window would display a Location (or address) bar.

You can use the action diagramming statements: Enable and Disable to control the display of the hypertext link. Alternatively, you can get the details of the conditions under which a control is disabled or enabled by checking the Disabled By or Enabled By options.

Display Push Buttons as Hypertext Links

A push button can be displayed as a hypertext link by selecting a check box, Display as Hypertext Link, on a Push Button Properties dialog. This check box is enabled in HTML mode and ASP.Net mode. This feature lets you perform functions associated with push buttons, such as Executes Click Event, while maintaining the appearance of a hypertext link.


Push Button Properties

Button
Text: add
Mnemonic Key:
Name: Button1
☐ Is the Default Push Button
☑ Display As Hypertext Link
Command... <NONE>

Bitmap
Bitmap: <NONE>
Number of Pictures: 1
☑ Adjust Button to Bitmap size
HTML Extension:

Accelerator
☐ Ctrl ☐ Alt ☐ Shift Key: <None>

Button Action
○ Special Action <NONE>
○ Initiates the Dialog Box <NONE>
● Executes the Procedure Step

Varying and Fixed Size Tables

HTML Tables are a type of list that can be associated with group views. ASP.NET Web Client application supports two types of HTML Tables:

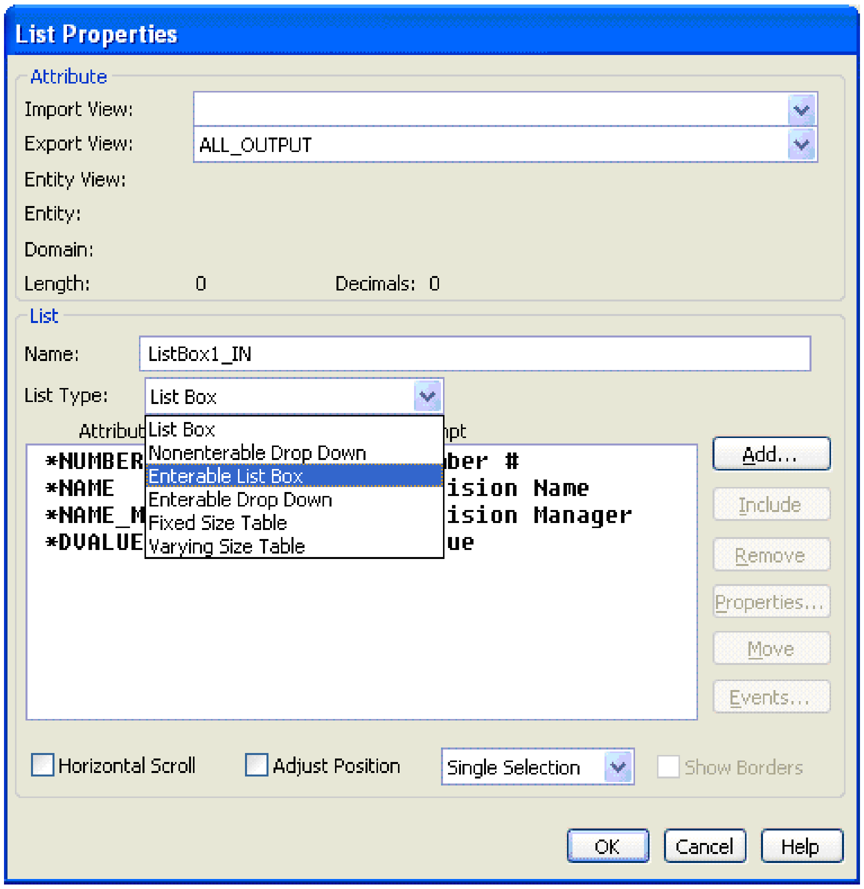
- Fixed Size Table

  The coordinates of a Fixed Size Table are specified at design time. A Fixed Size Table will only display the number of rows that can fit in the size designed for it. Any controls placed below a fixed size Table will retain their designed positions at Runtime.

- Varying Size Table

  The vertical size of a Varying Size Table is directly proportional to the size of its associated Group View. Varying Size Tables grow or shrink in terms of the number of rows they display to ensure all the entries in their associated group views are displayed. Any controls located below Varying Size tables move to maintain the designed distance between them and the Table.

The following screenshot displays the List Properties dialog:



Dynamically Change Multi-State Images

GUI applications have the ability to change images on push buttons according to their state, for example, disabled versus enabled. For GUI applications, a user specifies a bitmap that is divided into sections, each representing a different state. ASP.NET Web Client application provides a similar capability. Where for GUI applications you had a different section of the provided bitmap designated for each state, with Web application, you have to provide a different image in a different file for each state. The alternative images will use the image name in the model as a base and append text to it for the four different states supported: enabled, disabled, pushed, and focused. If myimage is the name saved in the model, the series of images used will be of the forms myimage.jpg for the enabled state, myimage_disabled .jpg for the disabled state, myimage_ pushed .jpg for the pushed state and myimage_focused .jpg for the focused state. The images should be placed under the html\images\ directory, and the value in the Number of Pictures field on the Push Button Properties dialog must be set to the number of images.

Support for Themes and Skins

CA Gen provides support for themes and skins in ASP.NET applications. Themes and skins let you create, and maintain common appearance settings across an ASP.NET application.

CSS Files

A theme may also include Cascading Style Sheet (CSS) files. When you add a *[set the File Name variable].css* file for a theme in the *<themename>* directory, the style sheet is applied automatically as part of the theme.

The CSS file consists of valid style sheet statements that specify the Type Selector, or the ID Selector for the windows or controls in a CA Gen ASP.NET application.

In CA Gen ASP.NET applications, the appearance settings of a control from the Navigation Diagram, dynamic attributes, Dot Notation statements, or skin properties are saved as inline style attributes in each HTML tag. Hence, all controls in CA Gen ASP.NET applications contain inline style attributes.

The style attributes in a CSS file can only modify the attributes that are not specified in the inline style attribute because the inline style attributes have a higher precedence over the style settings from a CSS file,

For example, the following statements in a CSS file define the background color of a window, the color of the field prompt, and the controls with ID, *Field1*:

```
Body { background-color:#ffc0cb; }
Label   {  color: #00ff00;  background-color: #32cd32; }
#Field1 { color:#ff0000; background-color: ffc0cb;}
```

If you add this CSS file in the *<themename>* directory, the following appearance settings apply for the windows, controls, and field prompts:

- All the windows in the application display with pink background color
- The field prompts in the application display with green foreground color, and lime-green background color.
- The controls with the ID, *Field1*, in the application display with red foreground color, and pink background color if the inline styles for the prompts and controls with ID, *Field1*, do not specify the foreground color or the background-color.

Image Files

A theme may include images that are referenced by the CSS file.

If the style statement of a CSS file references an image file, you must add the image file to the *<themename>* directory or the Images subdirectory under the *<themename>* directory depending on the style statement in the CSS file.

Note: In CA Gen ASP.NET application, the ImageUrl property is not supported for ASPXBitmap, ASPXBitmapButton, and ASPXHyperLink controls in the theme. The control skin must not reference the image files.

Support for Third-Party Web Controls

Third-party Web controls and their event management is supported. This lets ASP.NET Web Client add capabilities specific to the ASP.NET environment, through the use of external controls that are not part of the traditionally available set of Gen controls. These controls can be used in a manner similar to the way in which ActiveX controls are currently supported in GUI clients. Web Controls can be rendered using Internet Explorer without the need to install any third-party software on the end-user machine. However, Web Controls need to be installed on the Development, Build, and Application Server computers.

Toolset Support for Third-Party Web Controls

The Toolset lets you add third-party Web Controls to windows and dialogs. To do so, you need to first install the third-party Web Control of your choice on your development machine. You can then add the third-party Web Control to the model through the ASP.NET Control Manager.

Note: If you want to upgrade the version of a third-party Web Control in a model, you must remove the earlier version and add the new version in the Toolset using the ASP.NET Control Manager. As a result, the toolset updates the file, ThirdPartyControls.xml in the CA Gen directory. This file retains the controls' version information and must be updated when a third-party Web Control is upgraded.

Insert Third-Party Controls from the Navigation Diagram

You can insert third-party controls in a window or a dialog from the Navigation Diagram in the Toolset.

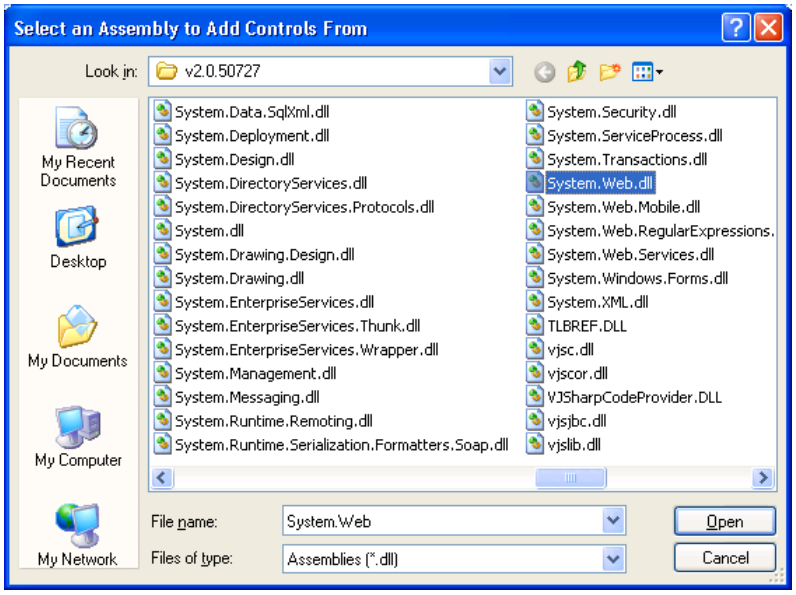
Follow these steps:

1. Click Edit, ASP.NET Control Manager in the Toolset.

   The Third-Party .NET Control Manager dialog opens.

2. Click Add on the Third-Party .NET Control Manager.

   The Select an Assembly to Add Controls From dialog opens.

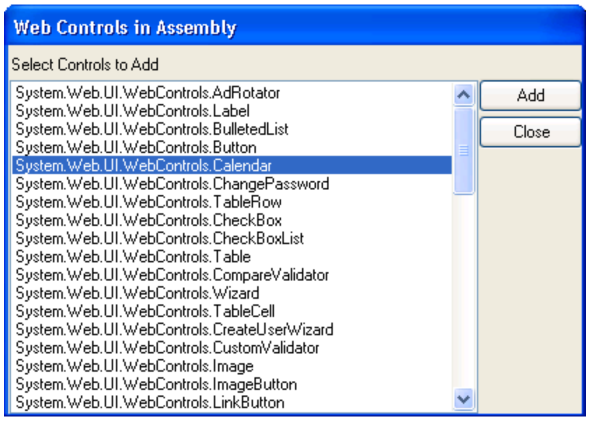


System.Data.SqlXml.dll
System.Deployment.dll
System.Design.dll
System.DirectoryServices.dll
System.DirectoryServices.Protocols.dll
System.dll
System.Drawing.Design.dll
System.Drawing.dll
System.EnterpriseServices.dll
System.EnterpriseServices.Thunk.dll
System.EnterpriseServices.Wrapper.dll
System.Management.dll
System.Messaging.dll
System.Runtime.Remoting.dll
System.Runtime.Serialization.Formatters.Soap.dll
System.Security.dll
System.ServiceProcess.dll
System.Transactions.dll
System.Web.dll
System.Web.Mobile.dll
System.Web.RegularExpressions.
System.Web.Services.dll
System.Windows.Forms.dll
System.XML.dll
TLBREF.DLL
vjsc.dll
vjscor.dll
VJSharpCodeProvider.DLL
vjsjbc.dll
vjslib.dll

File name: System.Web
Files of type: Assemblies (*.dll)

3. Select the required assembly, and click Open.

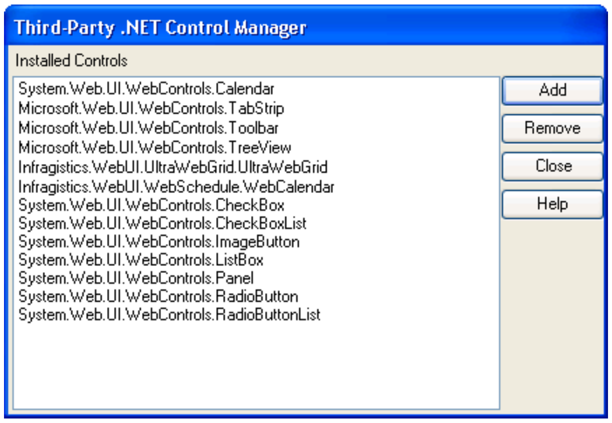
   The Web Controls in Assembly dialog opens.



System.Web.UI.WebControls.AdRotator
System.Web.UI.WebControls.Label
System.Web.UI.WebControls.BulletedList
System.Web.UI.WebControls.Button
System.Web.UI.WebControls.Calendar
System.Web.UI.WebControls.ChangePassword
System.Web.UI.WebControls.TableRow
System.Web.UI.WebControls.CheckBox
System.Web.UI.WebControls.CheckBoxList
System.Web.UI.WebControls.Table
System.Web.UI.WebControls.CompareValidator
System.Web.UI.WebControls.Wizard
System.Web.UI.WebControls.TableCell
System.Web.UI.WebControls.CreateUserWizard
System.Web.UI.WebControls.CustomValidator
System.Web.UI.WebControls.Image
System.Web.UI.WebControls.ImageButton
System.Web.UI.WebControls.LinkButton

4. Select the required controls in the Web Controls in Assembly dialog, and click Add.

   The selected controls appear in the Third-Party .NET Control Manager dialog, and the Web Controls in Assembly dialog closes.



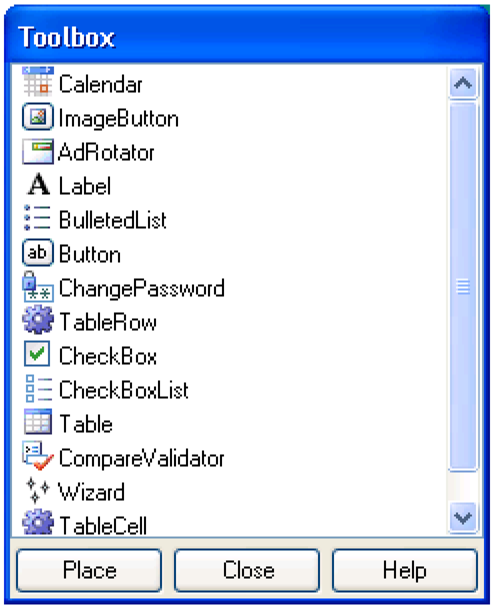
System.Web.UI.WebControls.Calendar
Microsoft.Web.UI.WebControls.TabStrip
Microsoft.Web.UI.WebControls.Toolbar
Microsoft.Web.UI.WebControls.TreeView
Infragistics.WebUI.UltraWebGrid.UltraWebGrid
Infragistics.WebUI.WebSchedule.WebCalendar
System.Web.UI.WebControls.CheckBox
System.Web.UI.WebControls.CheckBoxList
System.Web.UI.WebControls.ImageButton
System.Web.UI.WebControls.ListBox
System.Web.UI.WebControls.Panel
System.Web.UI.WebControls.RadioButton
System.Web.UI.WebControls.RadioButtonList

5. (Optional) To delete a control, select the required control, and click Remove on the Third-Party .NET Control Manager dialog. A confirmation dialog appears.
6. Repeat Steps 2 to 5 to add all the required controls from different assemblies.
7. Click Close on the Third-Party .NET Control Manager dialog.

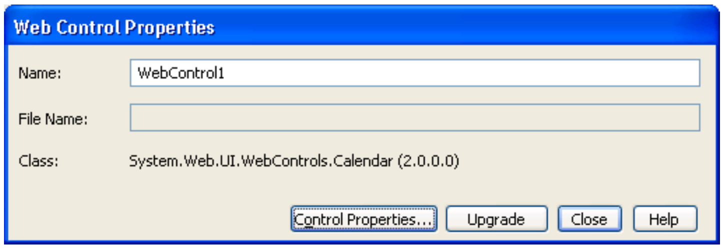
   The Third-Party .NET Control Manager dialog closes and returns to the window or the dialog that you were designing in the Navigation diagram.

8. Select Edit, Control Design from the Toolset to insert the required third-party control in the window or dialog.

   The selected third-party controls appear in the Toolbox.



9. Select the required control in the Toolbox, and click Place to position the third-party control on the current window, or dialog
10. Select the required control, and click Detail, Properties to modify the properties of the control.
11. The Web Control Properties dialog appears.



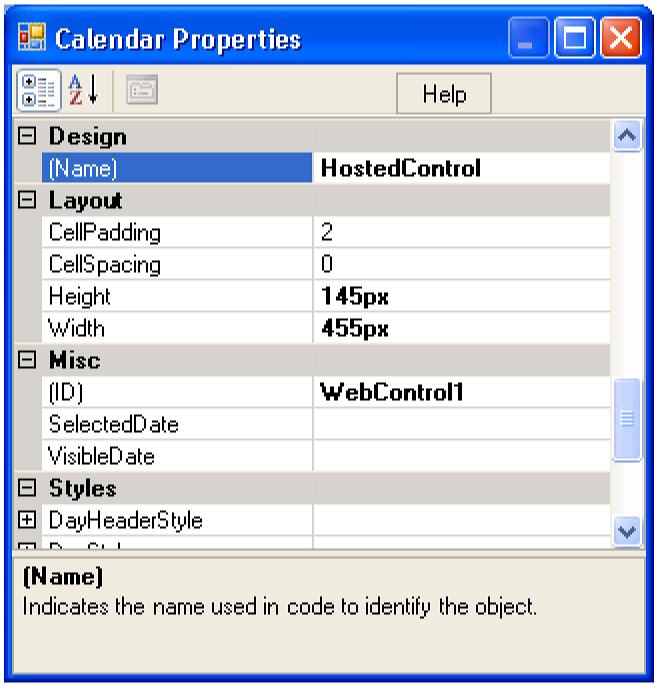
Name: WebControl1

File Name:

Class: System.Web.UI.WebControls.Calendar (2.0.0.0)

12. Select Control Properties on the Web Control Properties dialog.

    The Properties dialog appears for the control.



Design	
(Name)	**HostedControl**
Layout	
CellPadding	2
CellSpacing	0
Height	**145px**
Width	**455px**
Misc	
(ID)	**WebControl1**
SelectedDate	
VisibleDate	
Styles	
DayHeaderStyle	

(Name)
Indicates the name used in code to identify the object.

13. Specify the required modifications for the control in the Properties dialog, and close the dialog.

    The modified properties apply for the inserted third-party control at design time, and at runtime.

Access Third-Party Controls from Action Diagram

Action Diagram statements can be used to access methods and properties of third-party Web Controls in a similar fashion to how ActiveX controls are manipulated in the Action Diagram. Dot Notation plays a major role in this aspect. For more information about the Action Diagram, see the *Toolset Help*.

Restrictions on Use of Third-Party Web Controls

More Information:

Unsupported Features (see page 65)

Special Features

This section describes some of the special features of an ASP.NET Web Client application.

Browser Event Queue

Simultaneous events (within several hundred milliseconds), such as gainfocus/losefocus, are sent to the .NET Runtime in a single request. The events are sorted and processed in a similar manner as GUI clients. This event handling technique closely mimics GUI and lets users who are migrating from GUI to ASP.NET achieve maximum transparency with respect to migration. For multiple events, such as repetitive clicks, the browser's default event handling mechanism is used in conjunction with the ASP.NET framework's event handling mechanism. Note that CA Gen ASP.NET solution does not queue repetitive events on the browser.

Disabled By

Many applications disable controls depending on the values or states of other controls on the same page. Disabled controls appear unavailable and cannot be selected by a user of the application. Disabled By, which specifies the conditions under which a control is disabled, is supported. This facilitates porting GUI models to the Web.

DBCS Character Entry

Double-byte character set (DBCS) characters can be entered into non-DBCS fields without triggering an error unless you use edit patterns to verify entry data. Also, non-DBCS characters can be entered into a DBCS field without triggering an error unless edit patterns are used.

Specify Images Dynamically with SetBitMapName

In GUI applications during design, you can use the function SetBitmapName for a control to set the Bitmap on a picture control. You can also use the function SetBitmapName in ASP.NET applications. By using Dot Notation in the Action Diagram and invoking SetBitmapName you can set or modify a bitmap on its associated picture control. Unlike Web Generation applications, the specified bitmap needs to be manually included in an assembly by using the Additional File panel in the Build Tool. The specified bitmap has to be in the images folder.

Design Considerations

When creating the user interface for your client application, note that the toolset representation will not be identical to what you see at runtime. There are minor differences between what you see in different types of client applications: GUI, ASP.NET Web Clients, and Web Generation. Following are some of the differences you should expect, though the following items *do not* represent a comprehensive list of the differences between GUI, ASP.NET Web Clients, and Web Generation.

- Prompts in list boxes-ASP.Net Web Clients show prompts above the Table while in Web Generation they appear as part of the Table.
- Assembling of images-ASP.NET Web Client includes only images specified in the model. Web Generation includes all the contents of the HTML\images directory including subdirectories.
- Blank line option for dropdown list-ASP.Net Web Clients show blank line option in the Dropdown list while in Web Generation the <NONE> option is shown.

User Interface Refresh-In certain instances, ASP.NET Web Client applications will reflect updates on a page more frequently than GUI applications. For example, if in your application you update an export view and call a MessageBox, you would not see the window updated in GUI until you acknowledge the MessageBox. In ASP.NET Web Clients, you would see the update as soon as the MessageBox is displayed.

Chapter 5: Generating an ASP.NET Web Client Application

Generating ASP.NET Web Client components can be accomplished using either the Toolset or the Client Server Encyclopedia.

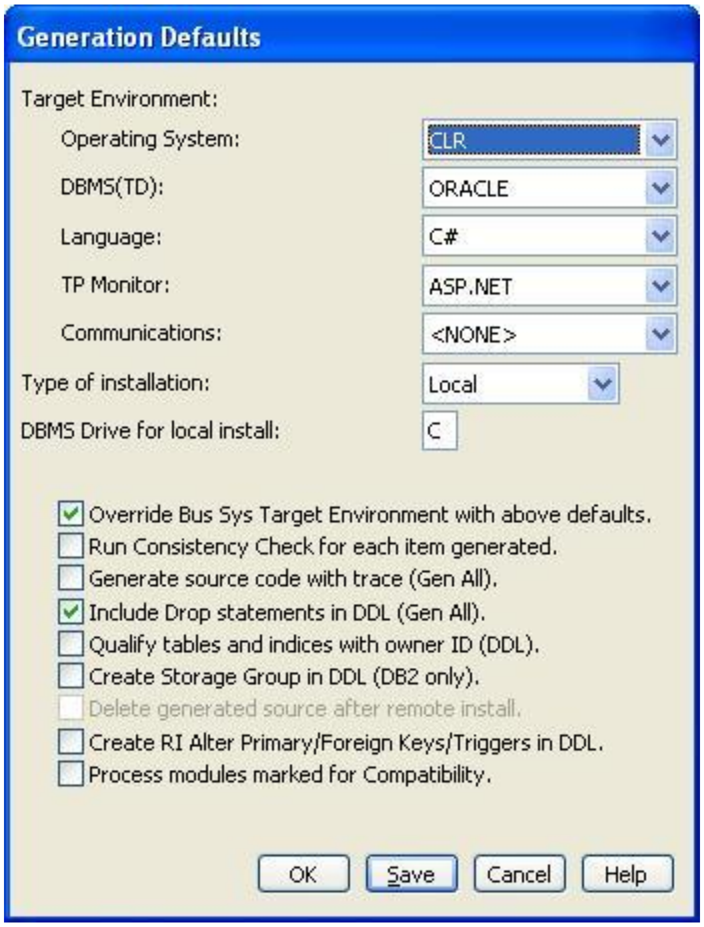
Generate ASP.NET Web Client Application from the Toolset

After the ASP.NET Web Client installation and design procedures are completed, you can generate ASP.NET Web Clients from the Toolset.

Follow these steps:

1. Start the Toolset and open your model.
2. From the Construction menu, select the Generation option.

3. From the Generation window, select the Options menu and then Generation Defaults. The Generation Defaults dialog appears:



Generation Defaults

Target Environment:

Operating System:	CLR
DBMS(TD):	ORACLE
Language:	C#
TP Monitor:	ASP.NET
Communications:	<NONE>
Type of installation:	Local
DBMS Drive for local install:	C

- [x] Override Bus Sys Target Environment with above defaults.
- [ ] Run Consistency Check for each item generated.
- [ ] Generate source code with trace (Gen All).
- [x] Include Drop statements in DDL (Gen All).
- [ ] Qualify tables and indices with owner ID (DDL).
- [ ] Create Storage Group in DDL (DB2 only).
- [ ] Delete generated source after remote install.
- [ ] Create RI Alter Primary/Foreign Keys/Triggers in DDL.
- [ ] Process modules marked for Compatibility.

4. Select CLR from the Operating System drop-down.

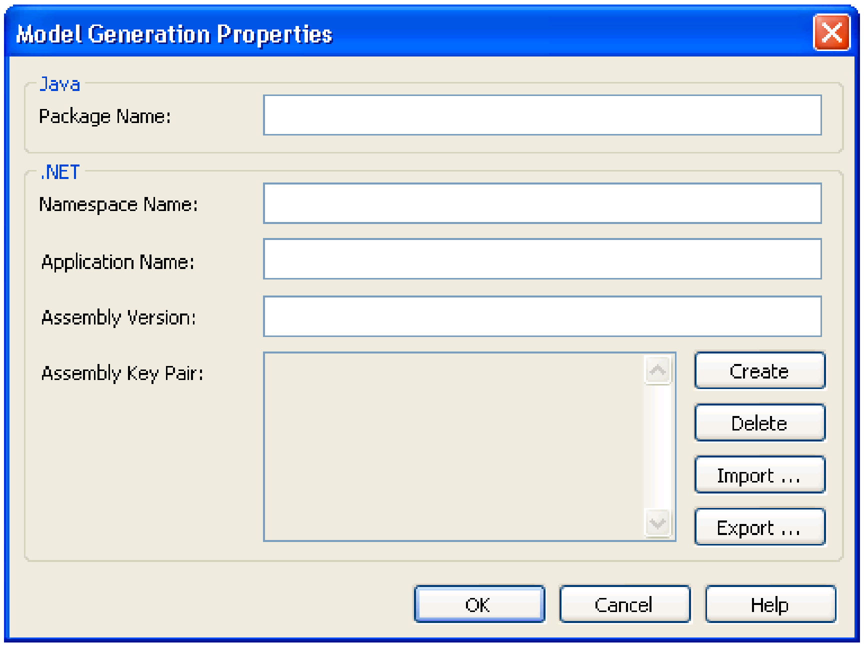
5. Select C# from the Language drop-down.

6. At the Type of Installation drop-down box, select Local or Remote.

   For a Remote install, generated files are packaged into one installation file, which can then be manually transferred to any Windows target system. For a Local install, generated files are compiled and prepared for installation onto a server. This server can either be running on the same machine as the toolset or a different machine. The default is Local.

7. Click OK.

8. Optionally, you can select the Options menu from the Generation window, then Model Generation Properties and enter the designated information in the .NET section.



9. Click OK and you are ready to continue with generation of an ASP.NET Web Client Application.

Consistency check rules identify the unsupported features in ASP.NET Web Client applications. These unsupported features are identified in the form of warnings. For a complete list, see the chapter Unsupported Features (see page 65).

A load module is considered eligible for ASP.NET Web Client generation if the ASP.NET Web Client settings are reflected in any one of the following:

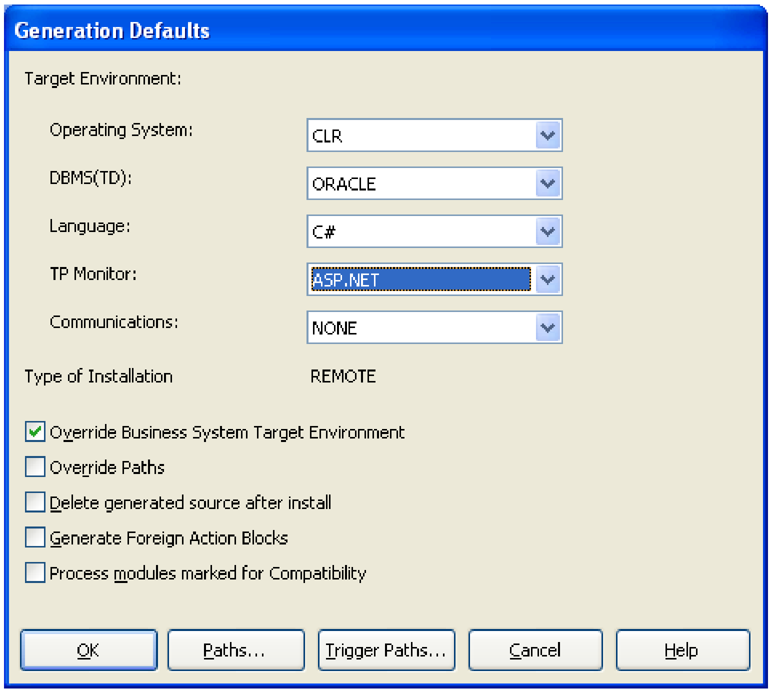
- Generation Defaults panel (when the Override Business System Target Environment with the preceding defaults check box is selected)
- Client Environment
- Business System Environment

Generate ASP.NET Web Client Application from CSE

You can also generate ASP.NET Web Client applications from the Client/Server Encyclopedia (CSE).

Follow these steps:

1. Start the Construction Client and open your model.
2. Select Code, Generate, Cooperative.
3. From Cooperative Application Generation, select Defaults; the following Generation Defaults dialog appears:


Generation Defaults

Target Environment:

Operating System: CLR

DBMS(TD): ORACLE

Language: C#

TP Monitor: ASP.NET

Communications: NONE

Type of Installation REMOTE

- [✓] Override Business System Target Environment
- [ ] Override Paths
- [ ] Delete generated source after install
- [ ] Generate Foreign Action Blocks
- [ ] Process modules marked for Compatibility

4. Select CLR from the Operating System drop-down.
5. Select C# from the Language drop-down.

After defining the ASP.NET Web Client components, you are ready to continue with the generation of an ASP.NET Web Client application.

Chapter 6: Building and Running an ASP.NET Web Client Application

Assembling an ASP.NET Web Client application means creating an MSI package that contains components necessary to execute the generated application. The MSI file is used to set up generated ASP.NET Web Client applications on IIS (Internet Information Services). To accommodate generating your ASP.NET Web Client application, the following directories are created under the model's local directory:

- C# directory

  Contains the generated source files.

- C#/build

  Building area for the generated model and contains the built dlls and web.config files.

- C#/deploy

  Contains the setup files, including the MSI file, for installing the generated application.

- xml directory

  Contains generated XML files.

- html directory

  Contains the html help files and the images directory

- html/images directory

  Contains converted images such as JPG, GIF, and PNG.

Images for ASP.NET Applications

Before installing the ASP.NET Web Client application, you should convert bitmap images (.bmp files) to a different format such as .jpeg, .jpg or .gif. Use a third-party tool (such as Netscape Composer or LView) to convert the images files to the format you prefer. Then copy the images into the images directory under the html directory. Since jpg is the default format assumed by the CA Gen generator, if a format other than jpg is chosen, you must set the HTML extension field on the Properties dialog from the Navigation diagram in the Toolset to match the file extension for the format you are using. Any images that are specified in SetBitMapName have to be manually included in the assembly images folder by using the Additional Files panel in Build Tool.

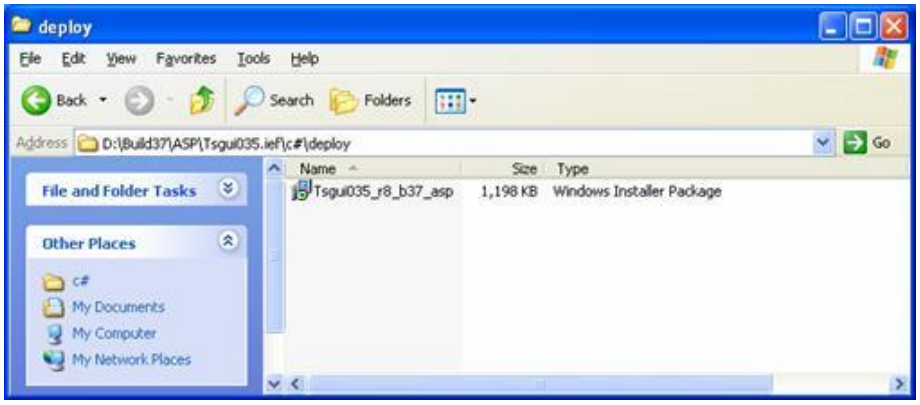
External Action Blocks

For more information about using External Action Blocks, see the *Distributed Processing-.NET Server User Guide*.

Build and Assemble ASP.NET Web Client Applications

After generating your application, you have to build it on a Windows platform. The Build Tool can work with either ICM files that result from a Local generation, or RMT files that result from a Remote generation. You can set options that affect the build process of your application through the Build Tool Profile.

The deploy directory <model.ief>\C#\Deploy, contains files required to install the application under Internet Information Services (IIS):



Executing the windows installer package installs your ASP.NET Web Client application under IIS.

During installation, MSI displays a warning message if a COM+ server with same name exists. You can ignore this message if the application targets Component Base Development (CBD).

Note: For more details about the Build Tool and Assemble utility, see the *Build Tool User Guide*.

Configure ASP.NET Web Client Communications

The ASP.NET Web Client runtime requires data to identify the CA Gen servers that it needs to establish communications with. There are two methods to configure communications. The first is through specifying the properties of each server in the CA Gen Toolset. The data is then saved in the model and results in the communication parameters being in the generated application. The second method uses the commcfg.txt file. Commcfg.txt is delivered as a sample file installed with CA Gen. It is a simple text file that you can place under your Application Server, or package within the application MSI file.

For more information about the Commcfg.txt file, see the *Distributed Processing - Overview Guide*.

Pre-compile ASP.NET Application

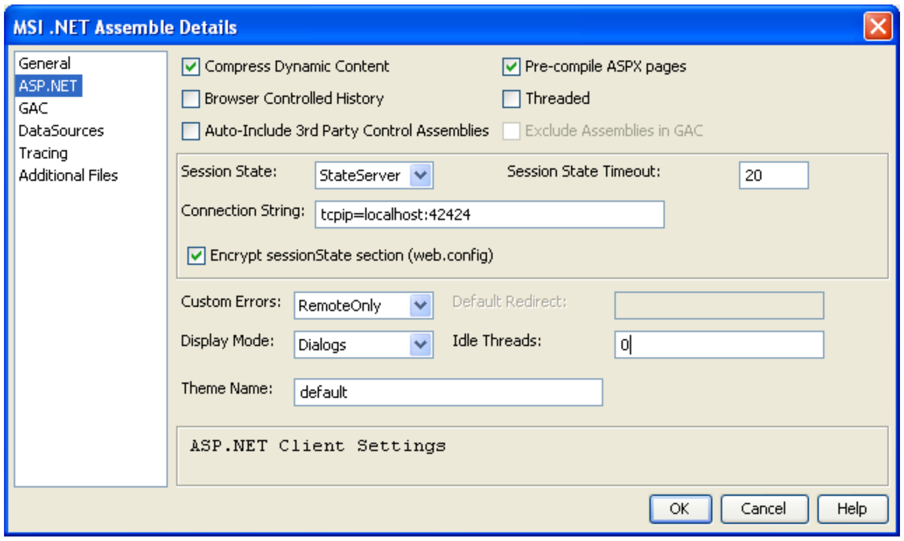
You can pre-compile an ASP.NET application before deploying it. Pre-compiling the applications has the following benefits:

- Improves application performance by avoiding the need to compile the first time the application is invoked from a browser.
- Lets you verify if the code in the application compiles without errors before executing the application.

Gen provides the in place mode for pre-compilation of ASP.NET applications. Pre-compiling a site in place effectively performs the same compilation that occurs when users request pages from your site. Therefore, the primary performance improvement is that pages do not have to be compiled for the first request. This feature is similar to the *Load JSP/Servlets at startup* feature that is available during assembly of a CA Gen Web Generation application.

With the MSI.NET Assemble Details dialog in the Build Tool, you can enable the pre-compiling of an ASP.NET Web Client application before deployment.

The following screenshot shows the MSI.NET Assemble Details dialog in the Build Tool:



☑ Compress Dynamic Content
☑ Pre-compile ASPX pages
☐ Browser Controlled History
☐ Threaded
☐ Auto-Include 3rd Party Control Assemblies
☐ Exclude Assemblies in GAC

Session State: StateServer
Session State Timeout: 20
Connection String: tcpip=localhost:42424
☑ Encrypt sessionState section (web.config)

Custom Errors: RemoteOnly
Default Redirect:
Display Mode: Dialogs
Idle Threads: 0
Theme Name: default

ASP.NET Client Settings

Note: For more information about pre-compiling ASP.NET Web Client application, and the MSI.NET Assemble Details dialog, see the *Build Tool Help.*

Trace Generated Applications

To debug and test your application you can enable tracing. This section explains how to enable tracing.

Trace Using web.config

The web.config file can be used to enable and disable tracing within the ASP.NET Web Client Runtimes. The output is written to a file, *trace-<appname>-<procid>.out*, where <*appname*> is the application name and <*procid*> is the process id of the application. This file is created in the %USERPROFILE%\AppData\Local\CA\Gen xx\logs\net directory.

To enable tracing using the web.config file, set the cmidebug value to -1.

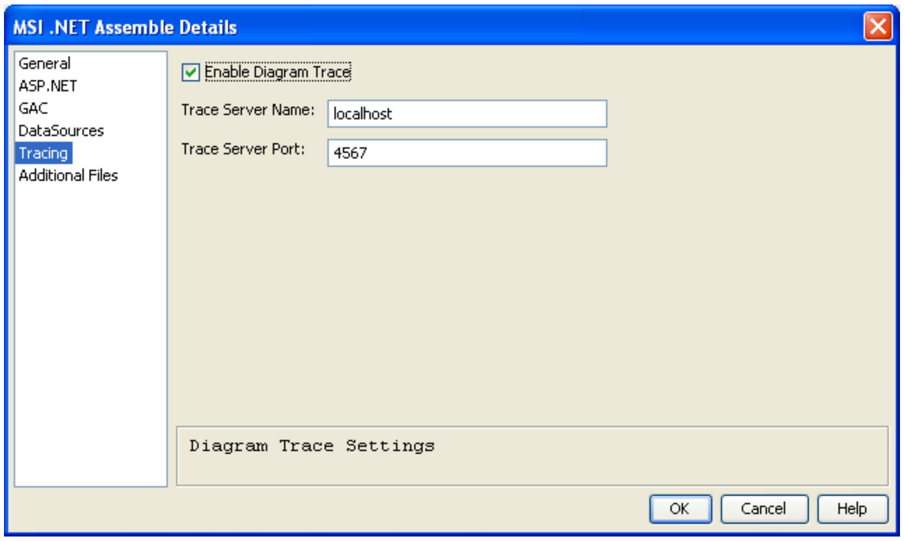
```
<add key="cmidebug" value="-1"/>
```

Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

Trace a Generated ASP.NET Application Using the Diagram Trace Utility

Follow these steps:

1. During generation, select to generate with Trace.
2. From the Build Tool, in the MSI .NET Assemble Details dialog, select Enable Diagram Trace under the Tracing node.



☑ Enable Diagram Trace

Trace Server Name: localhost

Trace Server Port: 4567

```
Diagram Trace Settings
```

3. To start the Diagram Trace Utility, click Start, All Programs, CA, Gen xx, Diagram Trace Utility.

   Note: *xx* refers to the current release of CA Gen. For the current release number, see the *Release Notes*.

   Note: Diagram Trace Utility executes only on Windows.

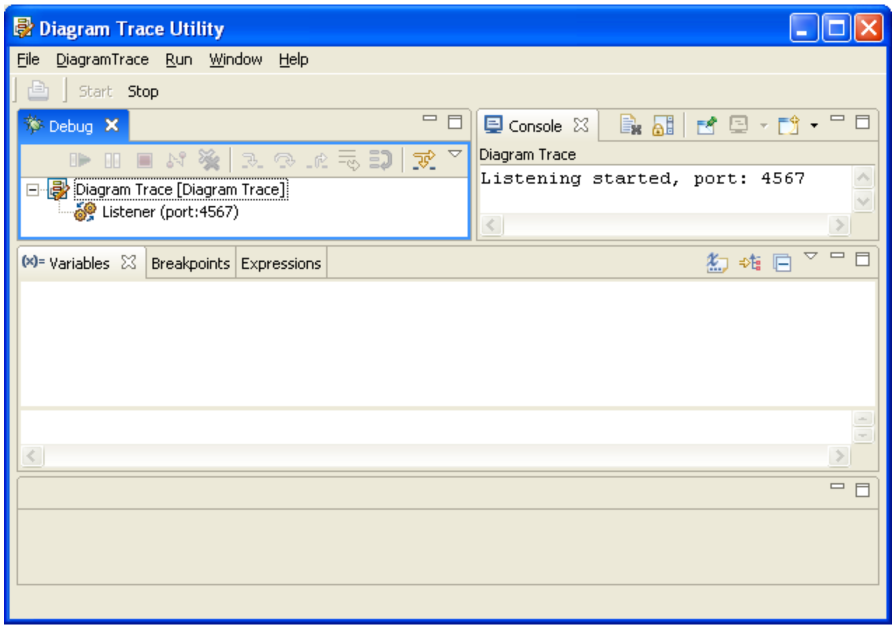


Diagram Trace Utility
File DiagramTrace Run Window Help
Start Stop
Debug
Diagram Trace [Diagram Trace]
Listener (port:4567)
Console
Variables Breakpoints Expressions

Diagram Trace

```
Listening started, port: 4567
```

4. The Diagram Trace Utility will *autostart* and listen on port 4567. You may change the default port from within the Diagram Trace Utility.
5. Start the application normally from the browser by invoking the ASP Handler (<trancode>.ashx).
6. Step through the Action Diagram code. For more information, see the *Diagram* Trace *Utility User Guide*.

Dynamic HTTP Compression

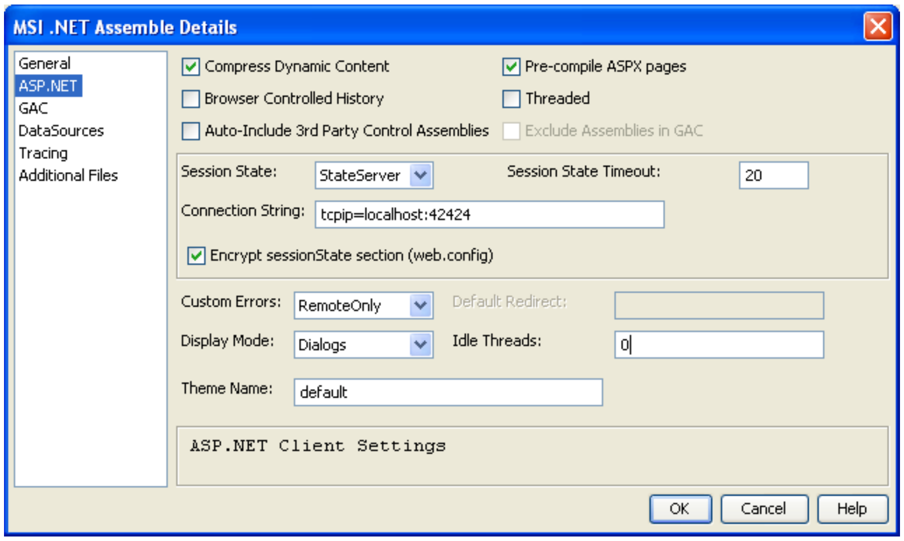
Compressing the content that is transferred across the network can improve application performance. Compression reduces the number of bytes of data that is sent over the network, resulting in faster transfer. The faster download outweighs the cost of expanding the compressed response at the browser.

Dynamic HTTP compression reduces the size of HTML and JavaScript content returned to the browser. Dynamic HTTP compressions use the GZIP compression technique. The dynamic content is sent through a filter and compressed on-the-fly by the CA Gen runtime.

Dynamic HTTP decompression is a feature built directly into the browser. This feature does not require any plug-ins, and is enabled by default.

For CA Gen ASP.NET applications, you can enable or disable dynamic HTTP compression from the MSI.NET Assemble Details dialog in the Build Tool.

The following image illustrates the MSI.NET Assemble Details dialog in the Build Tool:



- [✓] Compress Dynamic Content
- [✓] Pre-compile ASPX pages
- [ ] Browser Controlled History
- [ ] Threaded
- [ ] Auto-Include 3rd Party Control Assemblies
- [ ] Exclude Assemblies in GAC

Session State: StateServer
Session State Timeout: 20
Connection String: tcpip=localhost:42424
- [✓] Encrypt sessionState section (web.config)

Custom Errors: RemoteOnly
Default Redirect:
Display Mode: Dialogs
Idle Threads: 0
Theme Name: default

ASP.NET Client Settings

Note: For more information on dynamic HTTP compression, see the Build Tool Help.

Support for Native Image Generation

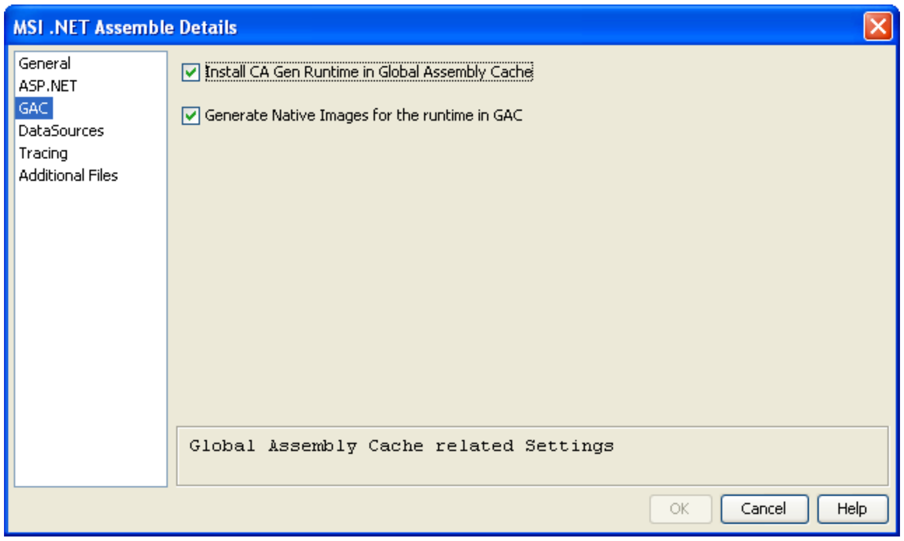
CA Gen supports native image generation of the Gen .NET runtime. Native images are files that contain compiled processor-specific machine code. The native images are installed in the native image cache on the local system. The .NET runtime can use the native images from the cache instead of using the Just-In-Time (JIT) compiler to compile the original assembly.

Native image generation lets assemblies load and execute faster because the .NET runtime reads code and data structures from the native image cache instead of dynamically generating code and data structures.

Note: To get the optimal performance from native images of Gen .NET Runtime assemblies, you must choose to install Gen .NET Runtime assemblies in the Global Assembly Cache (GAC) at the time of application assembly.

With the MSI.NET Assemble Details dialog in the Build Tool, you can enable the native image generation of Gen runtime for an ASP.NET Web Client application.

The following screenshot illustrates the MSI.NET Assemble Details dialog in the Build Tool:


MSI .NET Assemble Details

[✓] Install CA Gen Runtime in Global Assembly Cache

[✓] Generate Native Images for the runtime in GAC

Global Assembly Cache related Settings

Note: For more information on native image generation, and the MSI.NET Assemble Details dialog, see the *Build Tool Help.*

Support for Encryption

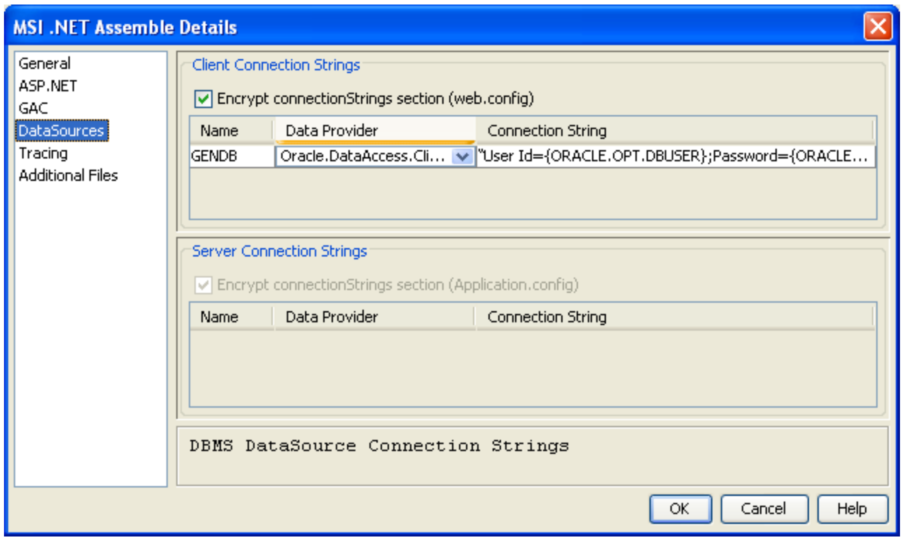
CA Gen stores the passwords of an ASP.NET Web Client application in the sessionState, and connectionStrings sections of the configuration files.

When building a CA Gen ASP.NET Web Client application, you have the option to encrypt the sessionState, and connectionStrings sections of the configuration files.

With the DataSources node on MSI .NET Assemble Details dialog in the Build Tool, you can enable the encryption of the following sections of the configuration files:

- The connectionStrings section in web.config for the Gen ASP.NET Web Client application.
- The connectionStrings section in Application.config for Component Services.

The following screenshot details the DataSources node on MSI .NET Assemble Details dialog:


MSI .NET Assemble Details

General
ASP.NET
GAC
DataSources
Tracing
Additional Files

Client Connection Strings

☑ Encrypt connectionStrings section (web.config)

Name	Data Provider	Connection String
GENDB	Oracle.DataAccess.Cli...	"User Id={ORACLE.OPT.DBUSER};Password={ORACLE...

Server Connection Strings

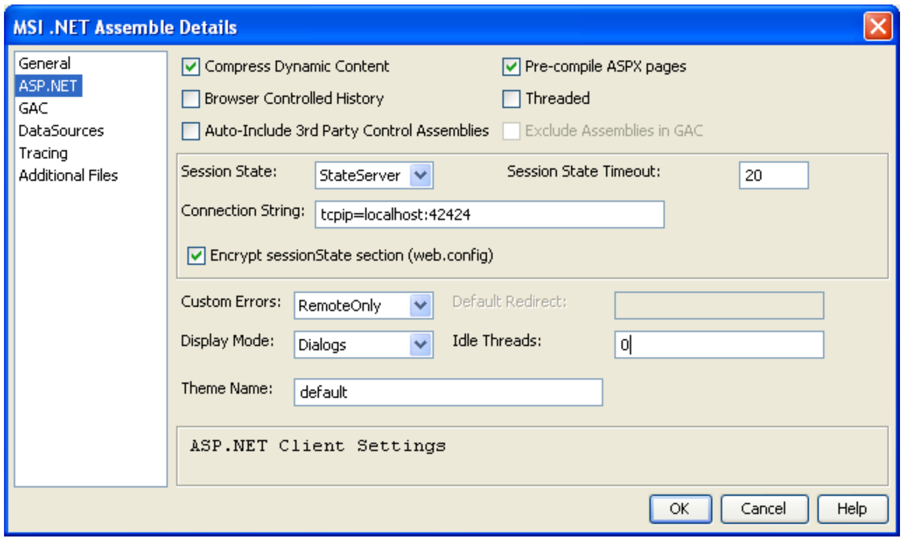
☑ Encrypt connectionStrings section (Application.config)

Name	Data Provider	Connection String

DBMS DataSource Connection Strings

You can also use the ASP.NET node on MSI .NET Assemble Details dialog in the Build Tool to encrypt the sessionState section in web.config for the client.

The following screeshot details the ASP.NET node on MSI .NET Assemble Details dialog:


MSI .NET Assemble Details

General
ASP.NET
GAC
DataSources
Tracing
Additional Files

☑ Compress Dynamic Content
☑ Pre-compile ASPX pages
☐ Browser Controlled History
☐ Threaded
☐ Auto-Include 3rd Party Control Assemblies
☐ Exclude Assemblies in GAC

Session State: StateServer
Session State Timeout: 20
Connection String: tcpip=localhost:42424
☑ Encrypt sessionState section (web.config)

Custom Errors: RemoteOnly
Default Redirect:
Display Mode: Dialogs
Idle Threads: 0
Theme Name: default

ASP.NET Client Settings

Note: For more information about the MSI .NET Assemble Details dialog, see the *Build Tool Help*.

More information:

Set Up the Application Server Environment (see page 18)

Edit Encrypted Sections of Configuration Files

After deployment, the user can edit the encrypted sessionState or connectionStrings sections in the application if the user has administrative privileges.

Edit the Encrypted sessionState Section in web.config

The user can edit the encrypted sessionState sections in web.config if the user has administrative privileges.

Follow these steps:

1. Decrypt the sessionState section by entering one of the following commands:
   ```
   aspnet_regiis -pdf system.web/sessionState <IIS Root>\myapp
   aspnet_regiis -pd system.web/sessionState -app /myapp
   ```
2. Edit the web.config file and make the required changes.
3. Re-encrypt the sessionState section by entering one of the following commands:
   ```
   aspnet_regiis -pef system.web/sessionState c:\inetpub\wwwroot\myapp
   aspnet_regiis -pe system.web/sessionState -app /myapp
   ```

Edit the Encrypted connectionStrings Section in web.config

The user can edit the encrypted connectionStrings sections in web.config if the user has administrative privileges.

Follow these steps:

1. Decrypt the connectionStrings section by entering one of the following command:
   ```
   aspnet_regiis -pdf connectionStrings c:\inetpub\wwwroot\myapp
   aspnet_regiis -pd connectionStrings -app /myapp
   ```
2. Edit the web.config file and make the required changes.
3. Re-encrypt the connectionStrings section by entering one of the following commands:
   ```
   aspnet_regiis -pef connectionStrings c:\inetpub\wwwroot\myapp
   aspnet_regiis -pe connectionStrings -app /myapp
   ```

Edit the Encrypted connectionStrings Section in Application.config

The user cannot directly edit the connectionStrings section in Application.config. However, the user can copy Application.config to a temporary file, rename the file as web.config, and make the required edits.

Follow these steps:

1. Copy the Application.config file to a temporary location and rename the file as web.config.
2. Decrypt the connectionStrings section in the temporary web.config file with the following command:
   `aspnet_regiis -pdf connectionStrings <web.config physical directory>`
3. Edit the temporary web.config file and make the required changes.
4. Re-encrypt the connectionStrings section in the temporary web.config with the following command:
   `aspnet_regiis -pdf connectionStrings <web.config physical directory>`
5. Copy the contents of the temporary web.config file to the original Application.config file.

Run ASP.NET Web Client

To invoke an ASP.NET Web Client application, an Http handler is used. This handler is of the form <trancode>.ashx or <loadmodule>.ashx. If the trancode and load module names are the same, <trancodeORloadmodule>.ashx will invoke processing for <trancode>.ashx. The load module can be explicitly invoked by specifying <loadmodule>_lm.ashx. Using <loadmodule>_lm.ashx invokes the default procedure step for the load module specified.

There is a web.config file that gets created with ASP.NET Web Client applications. This web.config contains mappings for Http Handlers to the runtime. The runtime infers the requested trancode from the mappings in web.config.

GUI applications allow you to define CLEAR SCREEN INPUTS at execution time, which affect the processing of the application. For ASP.NET Web Clients to use CLEAR SCREEN INPUTS, they must be sent on the URL as a parameter at request time:

```
<http://<hostname>/<application>/<trancode>.ashx? clearScreenInputs=myvalue
```

Support for Tabbed Browsing

CA Gen supports tabbed browsing in ASP.NET Web Client applications. Tabbed browsing lets the user concurrently open, access, and update an ASP.NET Web Client application from different tabs of a single browser instance.

A tabbed session is comparable to an individual browser instance of the ASP.NET Web Client application. CA Gen independently manages the session states of an application running under different tabs in a Browser Window.

Note: Tabbed browsing support requires a minimum of Internet Explorer 7.x.

National Language Support (NLS)

All files generated for ASP.NET Web Clients are UTF-8 encoded. The model encoding is used to determine the translation used to convert the files to UTF-8. In an effort to facilitate National Language Support (NLS), CA Gen creates satellite assemblies to represent each dialect. There are two ways to select the dialect:

- On Internet Explorer, the Language Preference setting under Internet Options is used to determine the corresponding default locale. There is a configuration file installed with CA Gen, culture.ini, which is used to determine mappings from Gen's standard Message Table designation in the Dialect Definition to Microsoft's standard locales.
- The dialect name or culture language code can be used as a context to reference the required dialect, for example:

  http://hostname:portnumber/application/dialect/trancode.ashx

  or

  http://hostname:portnumber/application/culturelanguagecode/trancode.ashx

Chapter 7: HttpRequest and HttpResponse Object Access

System.Web.HttpRequest and System.Web.HttpResponse are classes in the .NET Framework Class Library. HttpRequest lets ASP.NET read the HTTP values set by a client during a Web request. HttpResponse encapsulates HTTP response information from an ASP.NET operation. See Microsoft documentation for a list of the methods and properties of these classes.

Access Mechanism

This section explains the access mechanism provided by ASP.NET Web Client.

Access HttpRequest Object

ASP.NET Web Client provides a mechanism for accessing the HttpRequest object.

Follow these steps:

1. Design an External Action Block (EAB).
2. Use the EAB in the Pstep/Action Block where access to the requested information is required.
3. Design the views and attributes that will hold the request information.
4. Generate the EAB.
5. Add the following in the EAB method where user code is expected:

   // User-written code should be inserted here
   ```
   HttpRequest req = (HttpRequest)IefRuntimeParm2.GetRequest();
   ```
6. Query the required header such as:
   ```
   String userName = req.LogonUserIdentity.Name;
   ```
7. Set system attributes such as:
   ```
   Globdata.GetStateData().SetUserId(userName);
   ```
8. Set attributes in views such as:
   ```
   WOa.EUseridIefSuppliedCommand = userName;
   ```
9. Use the values set in the views in the application logic. Note that Gen global data (Globdata) is available here but CA strongly recommends that customers not use it, as it is subject to change.
10. Compile the application.

Use HttpResponse Object

Using methods and properties of the HttpResponse class is risky and could potentially cause runtime abends. For example, invoking the methods Redirect or Write, or WriteFile circumvents essential execution paths in the CA Gen runtime that impact history handling. Accessing certain Properties may also cause the application server to throw exceptions.

Follow these steps:

1. Design an External Action Block (EAB).
2. Use the EAB in the Pstep or Action Block where access to the response is required.
3. Design the views and attributes that hold the information needed to set the request.
4. Generate the EAB.
5. Add the following in the EAB method where user code is expected:

   ```
   // User written code should be inserted here.
   HttpResponse res = (HttpResponse)IefRuntimeParm2.GetResponse();
   ```

6. As an example, do the following to set a cookie:

   ```
   HttpCookie cookie = new HttpCookie("LastVisit");
   cookie.Value = DateTime.Now.ToString();
   cookie.Expires = DateTime.Now.AddDays(1);
   res.Cookies.Add(cookie);
   ```

7. Compile the application.

Chapter 8: User Exits

User exits are available with the various communication packages used.

Note: For more information about user exits, see the *User Exit Reference Guide*.

Chapter 9: Messages

This chapter provides a reference to the Consistency Check messages and Runtime Error messages.

Consistency Check Messages

New consistency check rules identify the unsupported features in ASP.NET Web Client applications. For a list of consistency check messages, see the following *Toolset* Help topics:

- Consistency Check AA00A - DZ99Z
- Consistency Check EA00A - GZ99Z
- Consistency Check HA00A - MZ99Z
- Consistency Check NA00A - RZ99Z
- Consistency Check SA00A - ZZ99Z

Runtime Error Messages

CA Gen ASP.NET Runtime provides messages that notify the user of a runtime error condition. The messages are stored in the assembly CA.Gen.localizationrt.dll and its satellite assemblies, CA.Gen.localizationrt.resources.dll, in the folder named with the specific locale for the satellite assembly. The assembly CA.Gen.localizationrt.dll contains the default English error messages, which are also used as the fall back resource when there is no localized message for the particular locale. The locale-specific error messages assembly CA.Gen.localizationrt.resources.dll contains the localized error messages for the specific locale. For example, CA.Gen.localizationrt.resources.dll in the %GENxx%Gen\.net\bin\es folder provides the Spanish localized error message.

CA Gen ASP.NET Runtime Error messages can be customized by recompiling the source files that are provided in the CA Gen installation. The source files are located in the %GENxx%Gen\.net\localization folder with the following file and folder structure:

```
%GENxx%Gen\.net\localization
 |-resources\:
 |    |-abrtmessages.es.txt
 |    |-abrtmessages.txt
 |    |-aspxmessages.ar.txt
 |    |-aspxmessages.da.txt
 |    |-aspxmessages.de.txt
 |    |-aspxmessages.es.txt
 |    |-aspxmessages.fi.txt
 |    |-aspxmessages.fr.txt
 |    |-aspxmessages.he.txt
 |    |-aspxmessages.it.txt
 |    |-aspxmessages.ja.txt
 |    |-aspxmessages.ko.txt
 |    |-aspxmessages.nl.txt
 |    |-aspxmessages.no.txt
 |    |-aspxmessages.sv.txt
 |    |-aspxmessages.txt
 |    |-csumessages.ar.txt
 |    |-csumessages.da.txt
 |    |-csumessages.de.txt
 |    |-csumessages.es.txt
 |    |-csumessages.fi.txt
 |    |-csumessages.fr.txt
 |    |-csumessages.he.txt
 |    |-csumessages.it.txt
 |    |-csumessages.ja.txt
 |    |-csumessages.ko.txt
 |    |-csumessages.nl.txt
 |    |-csumessages.no.txt
 |    |-csumessages.sv.txt
 |    |-csumessages.txt
 |    |-fmrtmessages.txt
 |    |-wfrtmessages.es.txt
 |    |-wfrtmessages.txt
 |    `-wmrtmessages.txt
 |-src\:
 |    |-AssemblyInfo.cs
 |    `-MessageBundle.cs
 |-localizationrt.snk
 `-make.bat
```

The TXT files in the resources folder contain the default and localized messages, and can be customized. The two C# source files in the src folder provide the assembly version information and the interface to be used by the ASP.NET runtimes to retrieve messages.

Rebuilding the Error Messages

The make.bat file is the batch file used to build the assemblies associated with the ASP.NET Runtime Error messages. As a prerequisite for building the assemblies, you must have Microsoft's C# compiler and .Net Framework installed on your system.

Follow these steps:

1. Launch an MS/DOS Command window.
2. Change your current directory to that which contains the make.bat file. Typically, this will be in the CA Gen install area, %GENxx%Gen\.net\localization subdirectory.
3. Refer to the make.bat file for execution instructions.
4. Run make.bat to build the ASP.NET Runtime Error message assemblies.

Redeploy the resulting assemblies for use with the appropriate applications.

Chapter 10: Unsupported Features

This chapter is intended to assist customers in converting their existing CA Gen Windows GUI clients to the new ASP.NET Web Client features as provided with CA Gen. The following features are unsupported or have partial support:

- Events
- Presentation-Related Functions
- OLE Functions
- Other Unsupported Features
- Window/Dialog Properties
- Window Controls
- Web Graphics
- Colors
- MAKE Support
- Help Support

Unsupported Events

The following CA Gen events are not supported by JavaScript and are ignored at runtime:

Note: Read-only entry fields do not intercept any events.

Control	Event Handlers
Window	RightMouseBtnDown RightMouseBtnUp WinMove WinResize
Single or multiline entry field	RightMouseBtnDown RightMouseBtnUp MouseMove Keypress
Check Box	RightMouseBtnDown RightMouseBtnUp MouseMove

Control	Event Handlers
Radio Button	RightMouseBtnDown RightMouseBtnUp MouseMove
Non-Enterable Dropdown	LeftMouseBtnDown LeftMouseBtnUp RightMouseBtnDown RightMouseBtnUp MouseMove
Non-Enterable List	RightMouseBtnDown RightMouseBtnUp MouseMove MouseEnter MouseExit GainFocus LoseFocus
Picture	RightMouseBtnDown RightMouseBtnUp MouseMove

Unsupported Presentation-Related Functions

The following CA Gen presentation-related functions are not supported by browsers:

- Beep
- ClearDisplayProperties
- GetHandle
- GetWidthMessageBoxBeep
- Mark Command
- Refresh
- RestoreDisplayProperties
- SaveDisplayProperties
- SetFontDynamically
- Unmark Command

Tips

- All the Get and Set (for example, GetHeight, SetTop) functions use Windows API calls that need the HANDLE to a specific control or a window displayed on the desktop to get or set a presentation property of that window or control. These types of Windows API calls are not possible in a browser environment.
- The functions Beep and MessageBoxBeep emit a beeping sound on the speaker of the computer where the code is running. Since browsers normally run on a different computer than the web server, these features are not useful in browser clients. MessageBoxBeep, however, produces a message box from the browser window. Only the audio beep is not supported.
- The functions ClearDisplayProperties, RestoreDisplayProperties, and SaveDisplayProperties work with the registry entries of the specified window's display properties and use Windows APIs that need the handles of the registry keys. Any changes made to the registry keys only affect the display properties associated with windows on that machine. For browser clients, who normally reside on a different computer, these functions do not provide the required effect.

  Additionally, some CA Gen Interface Object Methods and Properties are represented internally by Get/Set functions. Thus, they are not supported. The following CA Gen Interface Object Properties are not supported:

  – Enabled
  – Focus
  – Handle
  – Maximized
  – Minimized

  The following CA Gen Interface Object Methods are not supported:

  – Clear
  – Click (Check Box and Radio Button)
  – Copy
  – Cut
  – Paste
  – Redraw
  – SetBitmapBackground
  – SetSelection
  – Undo
  – Window.EnterableDropDownList
  – Window.EnterableDropDownLists

  - Window.EnterableListBox
  - Window.EnterableListBoxes
  - Window.OLEArea
  - Window.OLEAreas
  - Window.OLEControl
  - Window.OLEControls
  - OLE Functions

Unsupported OLE Functions

The following CA Gen OLE functions are not supported because Object Linking and Embedding (OLE) technology is not supported by browsers:

- PrintWindow
- Quit
- RegCloseKeys
- RegCreateKey
- RegDeleteKey
- RegDeleteValue
- RegEnumKey
- RegOpenKey
- RegQueryValue
- RegSetValue
- CreateObject

Unsupported File-Related Functions

The following CA Gen functions are used to create, view, or update files and documents on the local or network drives, and are not supported:

- OpenExcelDocument
- OpenWordDocument
- UpdateExcelCell
- ValueFromExcelCell

The functions OpenExcelDocument and OpenWordDocument cannot be supported because the client business logic runs on the Application server machine. Access to the file system of the browser's computer is considered a security violation. Also, some of these functions use OLE objects, which are not supported.

Even though file functions, such as OpenTextFile, are enabled in ASP.NET Web Client environments, they are not thread safe. Due to the inherent multi-threaded nature of Web Applications, care should be taken when using file functions. Since no locks are taken on resources accessed by file functions, upon accessing a file from a user session, other users will not be prevented from attempting to access the same resource. This poses the potential for collisions. Furthermore, references to these resources may not be preserved from one request to the next. You need to Open, Access, and Close within one event action because the information would not be preserved on a second execution if the thread changed. To avoid side effects and collisions between different sessions, you should create a unique file per user session. Where not supported, action diagram statements are ignored at runtime.

Other Unsupported Features

Due to the limitations imposed by HTML, the following controls are not supported:

- OLE Area
- ActiveX Controls
- Enterable DropDown List
- Enterable List

In addition, the following functionalities associated with specific controls are not supported:

- Auto tabbing among Entry Fields
- Extended selection in a Non-Enterable List
- Accelerators for PushButtons and Menu/MenuItems
- CBD - Cross model/application flows will only be supported if the same application name is defined for each component.
- Message boxes require threading to be enabled in the Assemble utility of the Build Tool. This option instructs the runtime to create a secondary thread for each request to be able to support MessageBoxes. Secondary threads cannot be serialized in Web Farms. Therefore, message boxes cannot be used in Web Farm environments.

Tips

- Tab sequencing among window controls is governed by the controls tab index in the generated XML file. Since a list box is implemented by HTML <TABLE>, sequencing does not include list boxes.
- Recursive calls within action blocks (such as an action block calling itself) are not supported and may result in runtime abends.

Window and Dialog Properties

The window's properties, such as initial position (Designed, Mouse Alignment, or System Placed) and style (System Menu, Minimize Button, Maximize Button, or Dialog Border) are not currently supported and such properties are ignored at runtime.

The user interfaces for a given window or dialog have a common appearance based on a predefined style sheet (Cascading Style Sheet).

The title property of the window or dialog is used that window or dialog. The icon file associated with the window or dialog is not supported and the property is ignored.

Web Graphics

All bitmaps must be converted to JPG, GIF, or PNG graphic formats. Background bitmaps are implemented by the background image style sheet property applied to the body element (and must be implemented by a JPG, GIF, or PNG graphic file).

Colors

An object's color properties stored in the model include the following:

- Background color
- Foreground color
- Highlighting background color
- Highlighting foreground color
- Disabled background color
- Disabled foreground color
- Background text window color
- Foreground text window color

The object's foreground color property in the model is used as the value for the color property in the style sheet. The object's background color property in the model is used as the value for the background color property in the style sheet. The other color properties are not supported and are ignored at runtime.

MAKE in Internet Explorer

The following table illustrates the CA Gen support for MAKE in Internet Explorer:

Control	**Color**	**Highlight Underline**	**Highlight Reverse**	**Intensity Dark**	**Error**	**Cursor**	**Protect**
Single-line edit field	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Multi-line edit field	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Check Box	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Radio Button	Yes	Yes	No	No	Yes	No	Yes
Dropdown List	Yes	Yes	Yes	Yes	Yes	Yes	Yes
List box	Yes	Yes	Yes	Yes	Yes	No	Yes

Considerations for Third-Party Web Controls

The following considerations are when using third-party Web Controls:

- ASP.NET Web Clients support Web Controls to the same level Visual Studio does. Problems identified in CA Gen ASP.NET Web Client must be proven to work in Visual Studio before CA can investigate them.
- Disable By is not supported for third-party Web Controls.
- Procedure Action Diagram access to Web Controls is limited to the current Procedure Step. That is, the GUI object representing the component cannot be passed using the views to another Procedure Step manipulation.
- Features that cannot be modified using methods and properties of a Web Control cannot be supported by ASP.NET Web Client. An example is specifying Sequencing of Tabbing among controls that are embedded on a Web control.
- Web Controls are only valid for a single window execution context. After another window replaces the current window (the window with the Web Control), the control is reinitialized on return. Additionally, any GUI Objects related to the control, which are held in the views, are then invalid. This restriction is dictated by the browser's caching mechanism and the stateless nature of Web applications.

- Validator controls are supported. You may add a Validator control to the Toolbox from the ASP.NET Control Manager, then select and place the Validator control from the Toolbox on to a web page. In CA Gen ASP.NET Web Clients, the name of a control becomes the ID of the control. The .Net Validator controls expect the ID of a control to be specified for validation. To specify which control to validate, type the name of the control to validate in the appropriate property of the Validator control (for example, in the RequiredFieldValidator control a property exists called ControlToValidate).

  Note: If you change the name of the control you are validating, you will also need to manually modify the Validator control's property.

  Container controls for ASP.NET Web Clients are an unknown entity at present. We have not found any ASP.NET Web Control that would be considered a “true” container control.

- Data-bound controls cannot be bound at design time. You can, however, associate them with data at runtime using the following techniques:

  1. Create the appropriate bound data type through the Action Block.
  2. Populate the data through the Action Block.
  3. Associate the data with the Data-bound control through the Action Block.

     Note: This requires a solid understanding of the Data-bound control and knowledge about how to create and populate the appropriate type for the Data-bound control.

Help Support

In Web applications, all help actions (Help, Help for Help, Extended Help, Keys Help, and Help Index) will display the generated loadModuleName_Help.html in a dialog.

Chapter 11: ADO.NET

ASP.NET Web Client is capable of supporting DBMS access from the action blocks running on the Application server machine. This type of access is made using ADO.NET calls through an ADO.NET datasource. Only those DBMSs specifically indicated in the *Technical Requirements* are supported. However, other DBMSs not specifically supported by Gen may be accessed using the ADO.NET/ODBC datasource if a proper ODBC driver exists for them.

To configure CA Gen to work with an ADO.NET datasource, you need the following information:

- ADO.NET datasource assembly
- Datasource
  - Target Database name,
  - Data provider
  - Connection string

DDL Installation

CA Gen does not directly support creating databases using ADO.NET. That leaves the user with the following options for installing the DDL into the target DBMS:

- Manual installation.
- With CA Gen, using the platform and DBMS specific generation options; for instance, generate for UNIX, Oracle and install with the CA Gen installation program.
- With CA Gen, using the ODBC/ADO.NET platform. When installing this way, the installation is performed by the Build Tool using an ODBC DDL installation program.

Note:

- There is an ODBC/ADO.NET DBMS in the Technical Design for naming objects that will be used in a DBMS that is not supported.
- If a DBMS will be used to support traditional CA Gen applications along with ASP.NET Web Clients or Java applications, matching the DBMS to get the table and column names is very important.
- When installing DDL using the ODBC approach, the actual database must already exist within the DBMS. ODBC drivers are incapable of performing any type of CREATE DATABASE command themselves.

Code Generation

To generate ASP.NET Web Client applications that perform ADO.NET access, use the following settings in the generation environment:

- OS-CLR
- DBMS-ODBC/ADO.NET: or a specific DBMS (see the DDL installation section for the reasons for selecting a specific DBMS)
- TPMONITOR-ASP.NET

Note: Even though the target OS is CLR, the actual build can only be performed using the Windows Build Tool.

Configure an ASP.NET Web Client Application for ADO.NET Access

You can configure CA Gen ASP.NET Web Client applications to support ADO.NET access from the Build Tool.

Follow these steps:

1. Install the DBMS vendor's ADO.NET datasource assembly, such that the target application can load the assembly by completing one of the following tasks:
   - Install the ADO.NET datasource assembly in the Global Assembly Cache (GAC).
   - Install the ADO.NET datasource assembly in a directory under the target application where assemblies are loadable by the application (in the application base directory or in an application private Path directory).
2. From the DataSources node on the MSI.NET Assemble Details dialog in the Build Tool, you must specify the following configuration information for datasources on the client and the server for each database:

   Name

   Specifies the name of the database that the application must access (as defined in the CA Gen model.)

   Data Provider

   Specifies the data provider for a datasource. You can select the datasource from the available list of supported databases, or you can enter a custom datasource in the enterable drop down list.

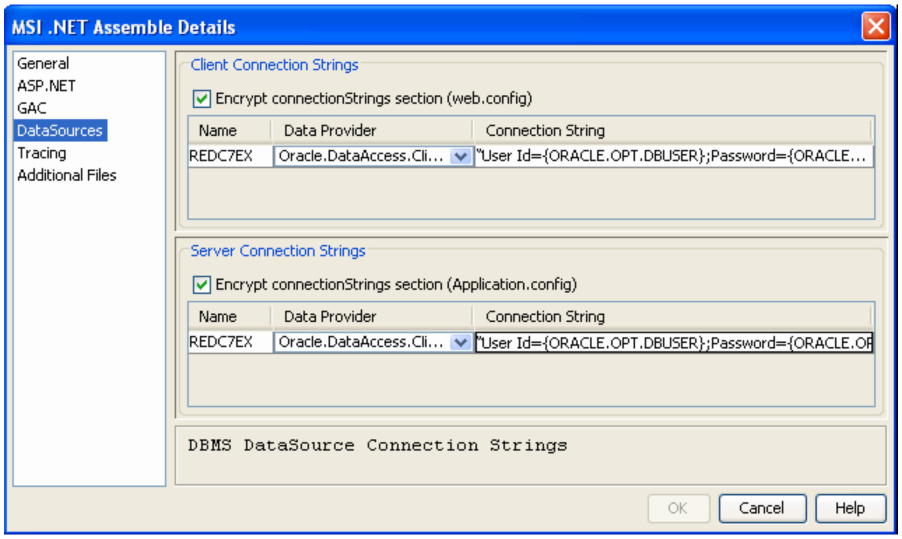
   Connection String

   Specifies the connection strings for the AD0.NET datasource. <Default> is applicable for the data providers that are available by default. If you specify a custom data provider, you must specify the connection string information for the data provider. You can also override the default connection strings settings for the supported data providers, and specify a custom connection string.

CA Gen constructs the default connection strings for the supported databases based on OPT.DBUSER, OPT.DBPSWD, and OPT.DBCONNECT profile tokens that are defined for the database in the Build Tool. CA Gen constructs the connection string information by default for the following supported data providers:

Data Provider	Connection String	Description
System.Data.Odbc	<Default>	Default data provider for generated ODBC and JDBC code.
IBM.Data.DB2	<Default>	Default data provider for generated DB2 code.
System.Data.SqlClient	<Default>	Default data provider for generated SqlServer code.
Oracle.DataAccess.Client	<Default>	Default data provider for generated Oracle code

The following screenshot illustrates the DataSources node on the MSI.NET Assemble Details dialog in the Build Tool:


MSI .NET Assemble Details
General
ASP.NET
GAC
DataSources
Tracing
Additional Files
Client Connection Strings
Server Connection Strings
OK
Cancel
Help

Client Connection Strings: [✓] Encrypt connectionStrings section (web.config)

Name	Data Provider	Connection String
REDC7EX	Oracle.DataAccess.Cli...	"User Id={ORACLE.OPT.DBUSER};Password={ORACLE...

Server Connection Strings: [✓] Encrypt connectionStrings section (Application.config)

Name	Data Provider	Connection String
REDC7EX	Oracle.DataAccess.Cli...	"User Id={ORACLE.OPT.DBUSER};Password={ORACLE.OP

```
DBMS DataSource Connection Strings
```

The DataSources node does not display if the ASP.NET Web Client application is not configured for database access.

Note: For more information about the MSI.NET Assemble Details Dialog, see the *Build Tool Help*.

Index