

CA Gen

Web Generation User Guide

Release 8.5



This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the "Documentation") is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with "Restricted Rights." Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

Copyright © 2013 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

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: Introduction	9
Web Generation	9
Web Generation Architecture.....	10
Web Generation Highlights	10
 Chapter 2: Pre-Installation Planning	 13
Technical Requirements	13
Supported Features	13
 Chapter 3: Installing Web Generation	 15
How to Install Web Generation	15
Set Up the Development Environment	15
Set Up the Build Environment.....	15
Set Up the Web Generation Application Server Environment	16
Set Up Message Resources.....	16
 Chapter 4: Designing a Web Generation Application	 19
Toolset Design Options	19
Common Edit Modifications.....	20
Window and Control Appearance.....	21
How to Specify Video Properties	22
Common Video Properties	23
Custom Video Properties for Java Web Generation Applications	23
Custom Video Properties for Standard Windows and Controls	24
Custom Video Properties for List Boxes and Tables	25
Cascading Style Sheets	29
Custom Video Properties for Message Boxes	30
Styles of Drop down Menus	31
Types of Hypertext Links.....	32
Display Push Buttons as Hypertext Links	34
Varying and Fixed Size Tables	34
Change Multi-State Images Dynamically	36
Embed ActiveX Controls.....	36
Restrictions on the Use of OCX Controls.....	36
Browser Prototype Mode	38

Consistency Check Rules	38
Design Considerations	38
Map ActiveX Controls Using HTML Editing	40
Map Submit and Reset Buttons	41
Web Generation Runtime Features	41
Using HTML Pages without Frames	41
Browser Event Queue	42
Support for ScrollTop and ScrollBottom Events	42
Applet-Free Approach	42
Enhanced List Box Scrolling and Extended Selection	42
Disabled By	43
Cookie-Free Approach	43
DBCS Character Entry	43
SetBitMapName to Specify Images Dynamically	43
Accessibility	44
Web Generation Additional Features	47
 Chapter 5: Generating a Web Generation Application	 49
Regeneration of Web Generation Applications	49
Generate a Web Generation Application from the Toolset	49
Generate a Web Generation Application from CSE	51
 Chapter 6: Building and Running a Web Generation Application	 53
Build and Assemble Web Generation Applications	53
Configure Web Generation Communications	54
Content Compression	54
Run Web Generation	55
Support for Tabbed Browsing	55
 Chapter 7: HttpServletRequest and HttpServletResponse Object Access	 57
HttpServletRequest	57
Access Method	57
HttpServletResponse	58
Access Method	59
 Chapter 8: Messages	 61
Consistency Check Messages	61
Web Generation General Error Messages 7000-7999	61
Web Generation Flow Manager Error Messages 7100-7199	62

Web Generation Servlet Manager Error Messages 7200-7299	63
---	----

Chapter 9: Unsupported Features **65**

Unsupported Events	66
Unsupported Presentation-Related Functions	67
Tips.....	68
Unsupported OLE Functions.....	70
Unsupported File-Related Functions.....	70
Other Unsupported Features	71
Window/Dialog Properties	72
Browser Differences	73
Web Graphics.....	73
Colors.....	73
MAKE Support.....	74
MAKE in Internet Explorer	74
MAKE in Mozilla Firefox	74
Asynchronous Support.....	75
Help Support	75

Chapter 10: JDBC Drivers **77**

DDL Installation	77
Code Generation	78
How to Configure the Target Environment	78

Chapter 11: Cross-Context Flows **81**

Cross-Context Flow	81
Using Cross-Context Flows	81
Performance Impact.....	82
Java Message Service (JMS) Session Storage.....	82
Map Load Modules	83
How to Configure a Cross-Context Flow	83
Design Considerations	84

Index **87**

Chapter 1: Introduction

The Web Generation feature of CA Gen uses the Web as an access medium to connect to CA Gen servers. The client interface is in the form of HTML and JavaScript that runs in a web browser. This makes it easy to use the services that are available on a server generated by CA Gen. It also makes the client application platform independent.

Users who want Web browser access to CA Gen models should benefit from reading this book. This guide describes design, generation, and building by providing the following information:

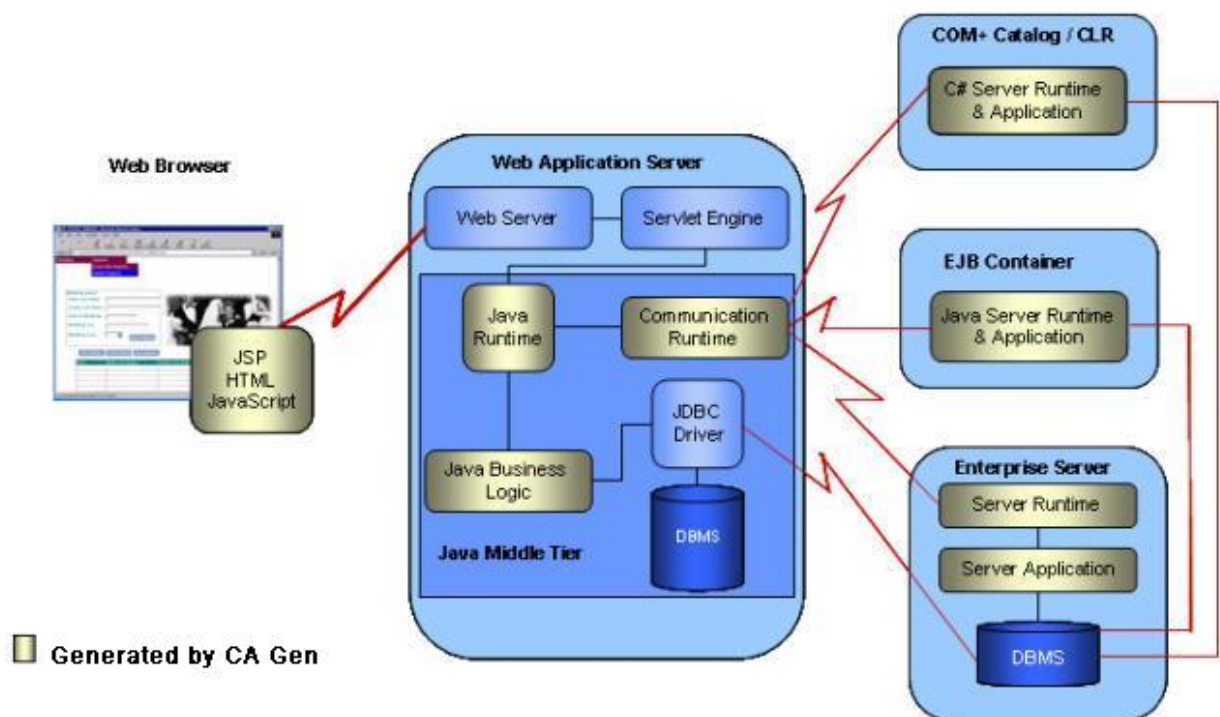
- What you need to know before installing the Web Generation software
- Technical requirements for Web Generation
- How to install the new software
- How to generate the application
- How to build files and use the application
- Consistency check messages that may display

Web Generation

The Web Generation feature lets a generated client, with an interface running on a browser, communicate with a CA Gen server. The Web Generation feature makes it easy for a client developer to use services available on a server. By using browsers as a client access medium, the Web Generation application makes CA Gen client's platform independent.

Web Generation Architecture

Web Generation lets users generate thin client applications, based on the GUI design model. The client application logic resides primarily on the Application Server, while the interface is downloaded to a web browser. The following illustration shows the high level architecture of a Web Generation application:



Web Generation Highlights

There are four environments involved in Web Generation that can be located on one machine or four physically separate machines. The following environments are involved in Web Generation:

- **Server environment**—The CA Gen runtime is installed for generated server applications. No special setup is required for Web Generation applications.
- **Application Server environment**—The runtime engine for the client is installed on an application server, which enables CA Gen to run the generated Web Generation application. The application servers must conform to the J2EE 1.4 standard.

- Development environment—The Toolset provides the environment, to develop and generate CA Gen Web Generation applications. This environment contains the components that generate the application both locally and remotely.
- Build environment—This environment enables the generated files from the CA Gen Development environment to be compiled and packaged into a Web Application Archive (WAR) and/or an Enterprise Application Archive (EAR). The CA Gen Build environment is often located on the same machine as the CA Gen Development environment.

The generated product includes the following highlights :

- The application logic resides on the application server machine.
- Java Server Pages (JSPs) serve as entry points to generated applications.
- The User Interface is a combination of HTML, JavaScript, and Cascading Style Sheets that is 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 Web Generation applications.
- The Web Generation software can access databases by using Gen Server applications or utilizing JDBC on the Application Server environment.
- The Web Generation client interface on Web browsers can be on any platform for which supported versions of Internet Explorer and Mozilla Firefox are available.
- JavaRMI, TCP/IP, MQSeries, Tuxedo, Webservices, and ECI are the only thread-safe middleware available for Web Generation.
- Due to browser and HTML limitations, some CA Gen related features have a different level of support when generated for Web Generation.

More information:

[Supported Features](#) (see page 13)

Chapter 2: Pre-Installation Planning

This chapter discusses the infrastructure requirements you must know as you prepare to install the Web Generation software. The four different environments involved in this feature can be located on one machine or four physically separate machines. The end-user machine must have a supported version of either Internet Explorer or Mozilla Firefox.

Technical Requirements

Third party software and CA Gen software each have requirements in addition to the CA Gen requirements as shown in the following table:

Environment	Third Party Software	CA Gen Software
CA Gen Server	No change required for Web Generation applications.	Installed CA Gen cooperative server.
Application Server Machine	The application servers must conform to the J2EE standards. **	Web Generation Runtime*. Communications Runtime**
Development Client	Windows Operating System **	Workstation Development Toolset or Client/Server Encyclopedia (CSE) for generation only. Web Generation.
Build Machine	Windows Operating System ** Java Platform, Standard Edition (Java SE)**	Implementation Toolset. Web Generation.

Note:

- * Only needed if not packaged in the application EAR file.
- ** See *Technical Requirements* available on <http://ca.com/support> for details.

Supported Features

Due to limitations of the browser and HTML, some features have a different level of support when generated using the CA Gen for Web Generation.

Chapter 3: Installing Web Generation

This chapter reviews the installation procedure for the Web Generation feature. Web Generation requires special instructions to set up the environment.

More information:

[Technical Requirements](#) (see page 13)

How to Install Web Generation

You must install Web Generation 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

To use the Web Generation application, you must set up your environment to generate and install the Web Generation component.

Set Up the Build Environment

The Build Tool has tokens that must be set to allow the generated applications to build on the implementation machine. You can set these using the Profile Manager in the Build Tool. The Assemble tool, accessible from the Build Tool's main panel, has options you can set to prepare your application for assembling. The Assemble Tool creates an EAR file containing your application in preparation for deployment to the Application Server.

Note: For more information, see the *Build Tool User Guide*.

Set Up the Web Generation Application Server Environment

If JDBC support is required on the web client, ensure that the Application Server has access to the JDBC drivers.

For an Application Server Environment machine, the Web Generation Runtime only needs to be installed if the server is going to reference the runtime through the CLASSPATH environment variable. If the application builder is going to use the Runtime in the EAR file, this step is not necessary.

Set Up Message Resources

Upon installation of the Java Web Generation application the message resource files are automatically installed in the <CA Gen install directory>\Gen\classes\resources directory. This list of files is not limited, but it must contain the following files:

- csmessages.java
- fmrtmessages.java
- smrtmessages.java
- wmrtmessages.java

These Localization files use the naming convention <manager type>messages + "_" + languageidentifier + ".java" where manager type is one of the four types (csu, fmrt, smrt or wmrt) and languageidentifier is a two letter code defined in the model under 'Dialect Definition' as the first two letters of the Message TableName entry.

All message files are Java source files stored in UTF-8 encoded Unicode. Unicode allows all the message files to be edited and viewed at the same time by a Unicode capable editor, for example, Microsoft Software Development Editor.

The message files must be compiled before they can be used in the runtime jar. The utility MKJAVART in <CA Gen install directory>\Gen\classes is used to compile the message files and build a runtime jar.

To build the runtime jar

1. Open a command prompt window and set the current directory to <CA Gen install directory>\Gen\classes.
2. Execute the MKJAVART batch file from the command prompt.
3. Enter the following in the command prompt to make the runtime for Web Client, with TCP/IP transport and the new message files:

```
mkjavart rt.jar WCE TCPIP
```


Rules for Editing Message Resource Files

The following syntactic rules apply when editing the message resource files.

- All entries are set using the method `ht.put("id", "message")` where *id* is the numeric (or text) ID of the message, and, *message* is the actual localized message text.
- All substituted variables in the message are indicated by curly braces {*n*}, where *n* is the numerical index of the data in the parameter array passed to the message formatting utility.
- Use `/* ... */` for all comments.
- Use a UTF-8 Unicode capable editor, for example, Microsoft Software Development Editor. Do not use Notepad.

Note: When updating messages, it is necessary to rebuild the runtime or re-assemble the application to include the changes in the application environment.

A re-assembled application must be redeployed to the application server to make the changed message classes available.

When using a global runtime for all applications, the new runtime must be redeployed to the application server, and the application server must be reset to read the new runtime.

Chapter 4: Designing a Web Generation Application

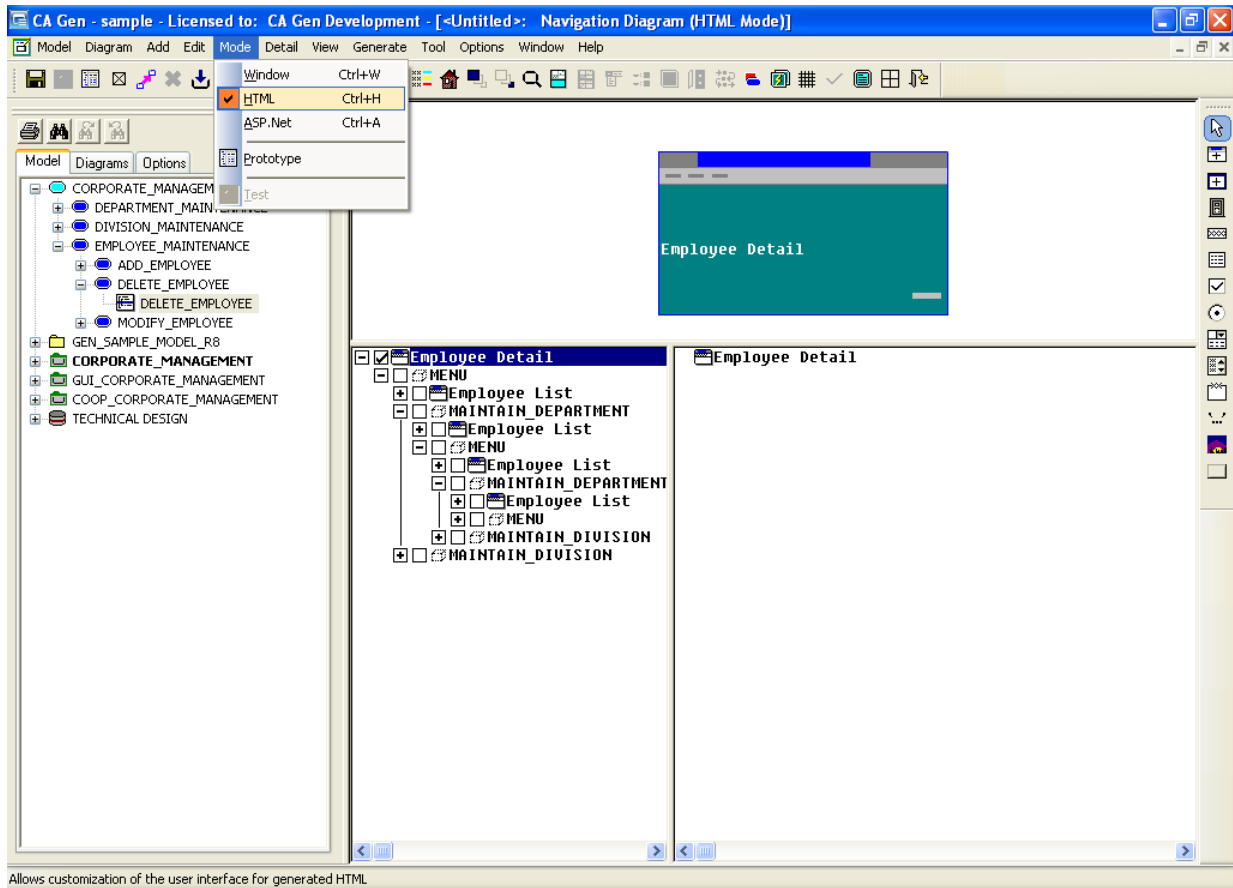
After installing the software, you are ready to use the Toolset to define Web Generation components.

This chapter contains information about design options available for Web Generation applications. Features that can influence the user interface of a Web Generation client are available through the Toolset and the Assemble utility of the Build Tool, which lets you use external Web Authoring tools with CA Gen models.

Toolset Design Options

The GUI Window design is the base user interface inherited by Web Generation applications. This facilitates using existing GUI or client/server models as a starting point for developing Web Generation applications. The Toolset allows the design to either remain common or to be specialized for different target user interface environments. Currently the supported options are: Window, HTML, and ASP.NET Edit Modes. The HTML Edit Mode is used for designing Web Generation applications.

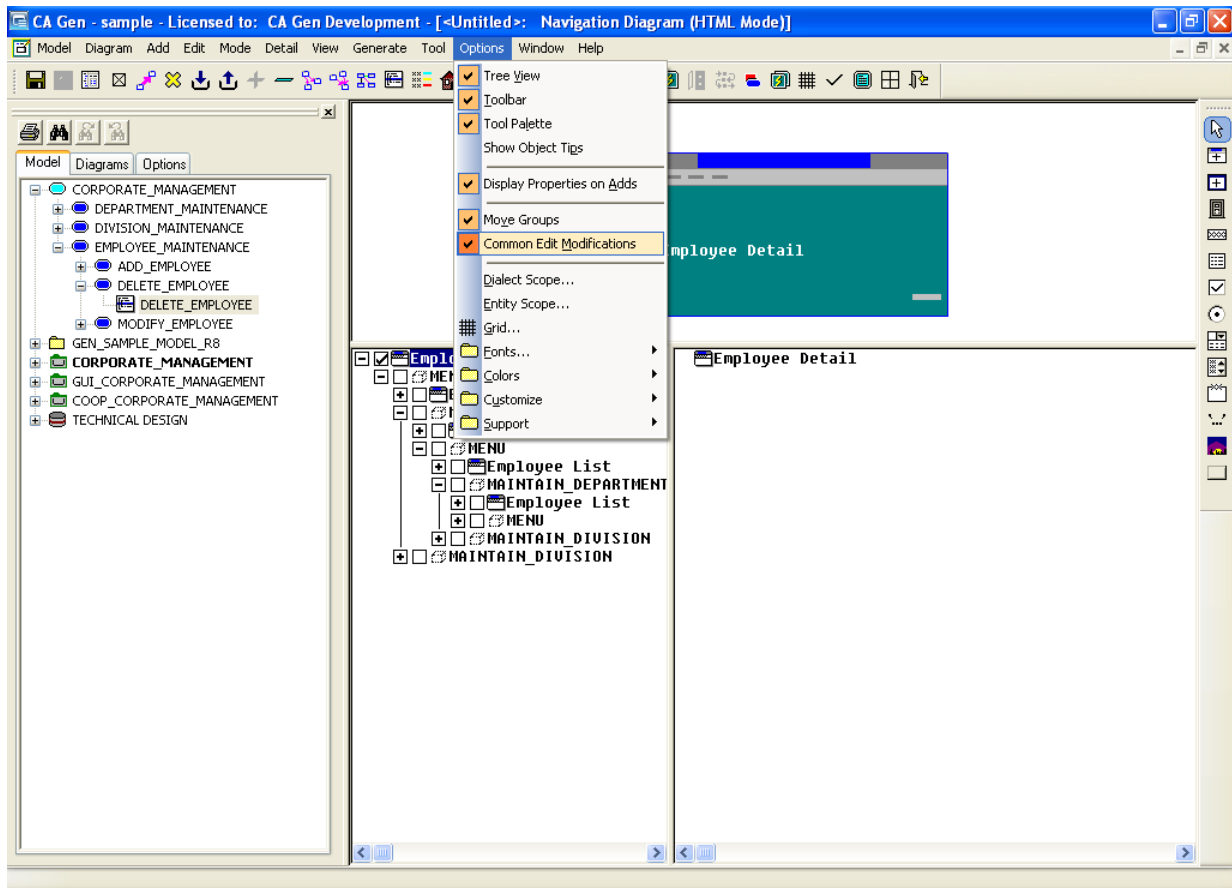
The following image illustrates how to navigate to the HTML Edit Mode in the Toolset:



Common Edit Modifications

The Common Edit Modifications menu item lets you make any edits to a window to be common to the three modes of the window: HTML, ASP.NET and Window modes. Moving a control in Edit Window mode moves the field when the window displays in Edit HTML mode and Edit ASP.NET mode. The same result is true when editing a field in either Edit HTML or ASP.NET modes. This common functionality is the default state of the Common Edit Modifications menu item and occurs when the item is checked.

The following image illustrates how to navigate to Common Edit Modifications on the Toolset:



If the menu item is not checked, then any edits causing updates to any of the visible properties of the windows and their controls are considered a customization for whichever edit mode the window is currently set (Window, HTML, or ASP.NET). Control location, prompt text, control font, and color are examples of customizable properties.

Note: Customizing for different modes will create additional objects in your model.

Window and Control Appearance

You can regulate the appearance of a window or control by using the Toolset's Business System Video Properties panels. For Web Generation applications, you have an additional method of regulating the appearance of windows and controls using custom video properties that are implemented as CSS Class attributes for the windows and controls. You can specify CSS Class attributes dynamically at runtime and statically during the design phase.

Both static and dynamic methods of specifying CSS Class attributes have benefits. When you specify CSS Class attributes during the design phase, you can adhere to strict style guides that define how a control or window should look throughout an application. Your design team can also roll out changes to the appearance of a control or window without having to modify an existing application.

By having the flexibility to dynamically specify CSS Class attributes at runtime, you can change the appearance of one control on one page. For example, you can change a `WindowObj.EntryField.CSSClass` to `pleasefillout` so the user sees a data entry field with a special appearance (like a red border and a bold font).

In addition to specifying Custom Video Properties (CSS Class attributes) to be used by all windows or controls within an application or assigning the Custom Video properties to one window or control, you can also specify a style scheme for a set of controls. For example, you can specify a Row/Column style scheme for all list boxes and tables, a style scheme for all message boxes, or a style scheme for all group box lines within your application.

How to Specify Video Properties

You can regulate the video properties of windows and controls by specifying how they will all look by default or by customizing the appearance of a window, a control, or a set of controls. Specifying default video properties provides a common look and feel for windows and controls throughout your application. Customizing the appearance of a window, control, or set of controls gives you the flexibility of changing the default to accommodate specific situations.

The video properties have the following order of precedence:

- Specify the default video properties for windows and controls in the Business System by selecting the Window option.
Note: For more information about setting default video properties or for setting a style scheme for a set of controls, see the *Toolset help*.
- Override or replace the default video properties set in the Business System for a window or control using the Navigation Diagram.
Note: For more information about overriding or replacing default video properties, see the *Toolset Help*.
- For web applications (Java Web Generation and ASP.NET Web applications) only:
Specify attributes that affect the appearance of the Web pages in the Business System by selecting the Web option.
Note: For more information, see the *Toolset Help*.

- For your Java Web Generation applications only:
 - You can dynamically change a window's or control's CSS Class property using Dot Notation statements in the Action Diagram.
Note: For more information about changing CSS Class properties dynamically, see the *Toolset Help*.
 - You can specify custom cascading style sheets, video properties for all standard CA Gen controls including GroupBox Lines, list boxes and tables, and video properties for message boxes. You can do so at the Business System level or for each instance of a control through the Navigation Diagram. You can also specify an external file containing cascading style sheet information at the Business System level by selecting the Java Web Generation option on the video properties panel.
Note: For more information about specifying cascading style sheets and video properties for your web applications, see the *Toolset Help*.

Common Video Properties

Common video properties are appearance settings that you can apply for a specific window or control. These properties *override* the default appearance settings for a window and a control in the Business System. With the Common Video Properties dialog in the Navigation Diagram of the Toolset, you can specify the foreground color, the background color, and the font characteristics for a specific window or control.

Note: For more information about setting common video properties, see the *Toolset Help*.

Custom Video Properties for Java Web Generation Applications

Custom video properties are appearance settings that you can define, such as background color, foreground color, font, and other style strings. You can define the custom video properties from the Custom Node of the Business System Defaults Video Properties Dialog. These appearance settings can be referenced by all the windows and control types in the Business System.

In the Navigation Diagram, you can specify the appearance setting for an individual instance of a window or a control by assigning a custom video property to a specific window or control. This setting from the Navigation diagram during the design phase is a static CSS Class setting. You can also specify custom video properties dynamically at runtime via Dot Notation statement. Before assigning a custom video property for windows and controls in an application, you must define a list of custom video properties.

The custom video properties generate the class selector style statements in the `<Business_System_Name>.css` file for the Java Web Generation application. The name of the custom video property is used as the CSS Class name in the generated style sheet statement.

When a custom video property is assigned to a window or a control, this property *replaces* the default appearance settings for a window and a control in the Business System.

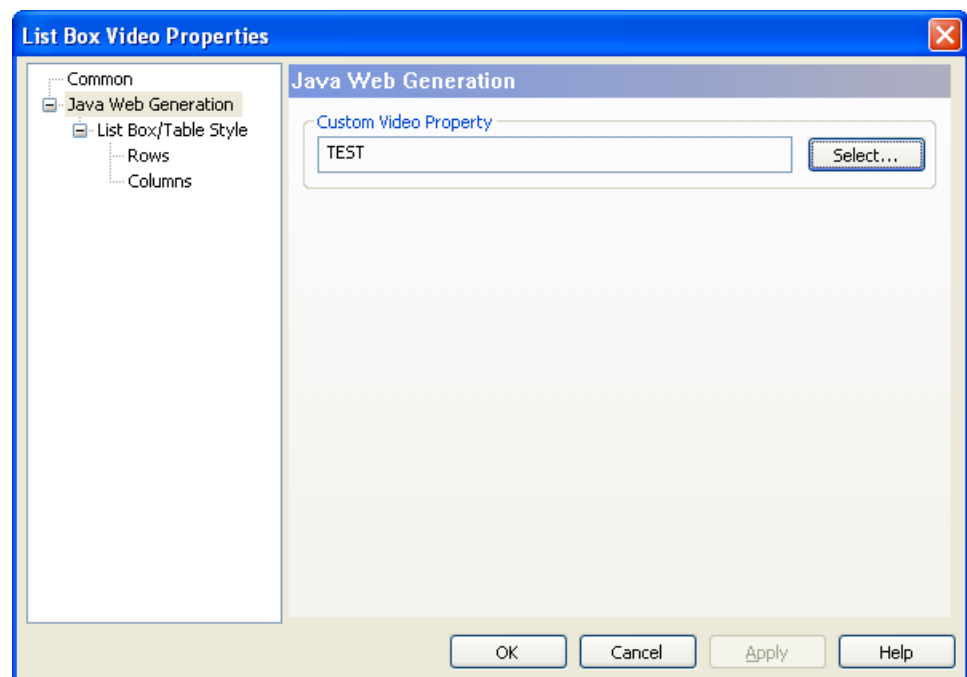
Note: For more information about specifying custom video properties for Java Web Generation applications, see the *Toolset Help*.

Custom Video Properties for Standard Windows and Controls

By assigning a custom video property to all the windows or controls in a Java Web Generation application, you can ensure that their appearance is consistent throughout an application. The predefined set of custom video properties let you specify a list of appearance settings, from which you can specify the required appearance setting for each window or control type at design time or at runtime.

With the Java Web Generation node in the Navigation Diagram Video Properties dialog, you can select a custom video property to replace the default video property for a specific window or control in a Java Web Generation application.

The following image illustrates the Java Web Generation node in the Navigation Diagram Video Properties:



You can also dynamically assign a custom video property or the default video property to a specific window or control with the *CssClass* and *PromptCssClass* properties by using Dot Notation statements.

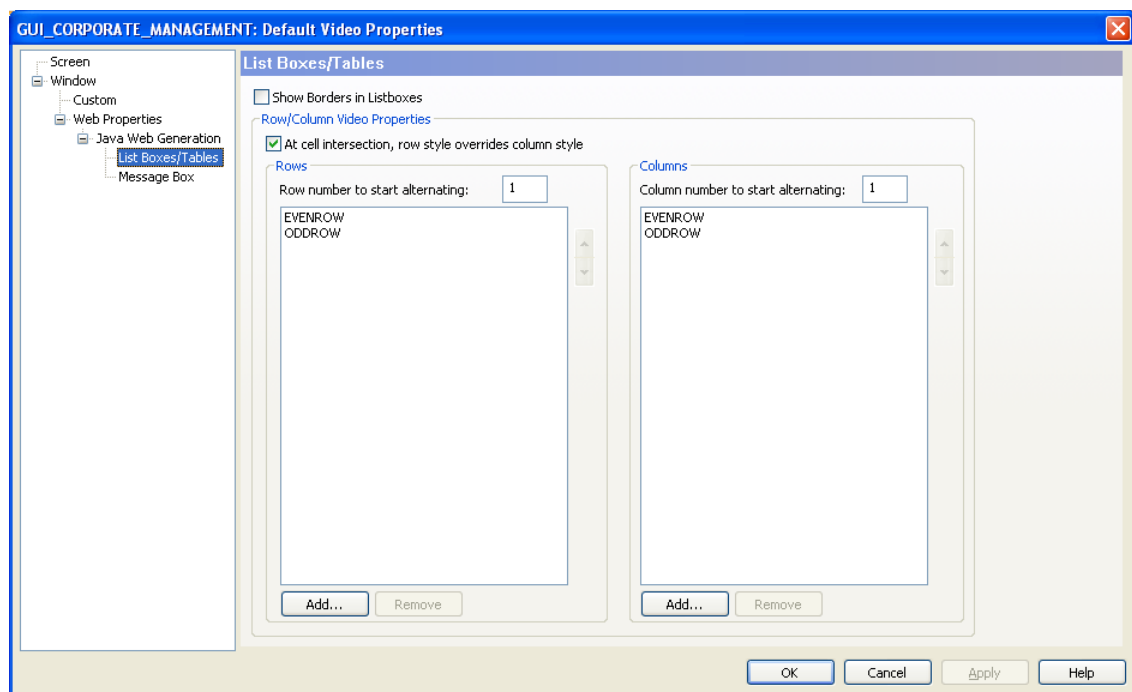
Note: For more information about specifying custom video properties for the standard windows and controls in Java Web Generation applications, see the *Toolset Help*.

Custom Video Properties for List Boxes and Tables

You can specify the appearance settings for all list boxes and tables by assigning a custom video property to these controls at the Business System level using the Video Properties dialog. Using this dialog, you can also specify the following characteristics for rows and columns in list boxes and tables:

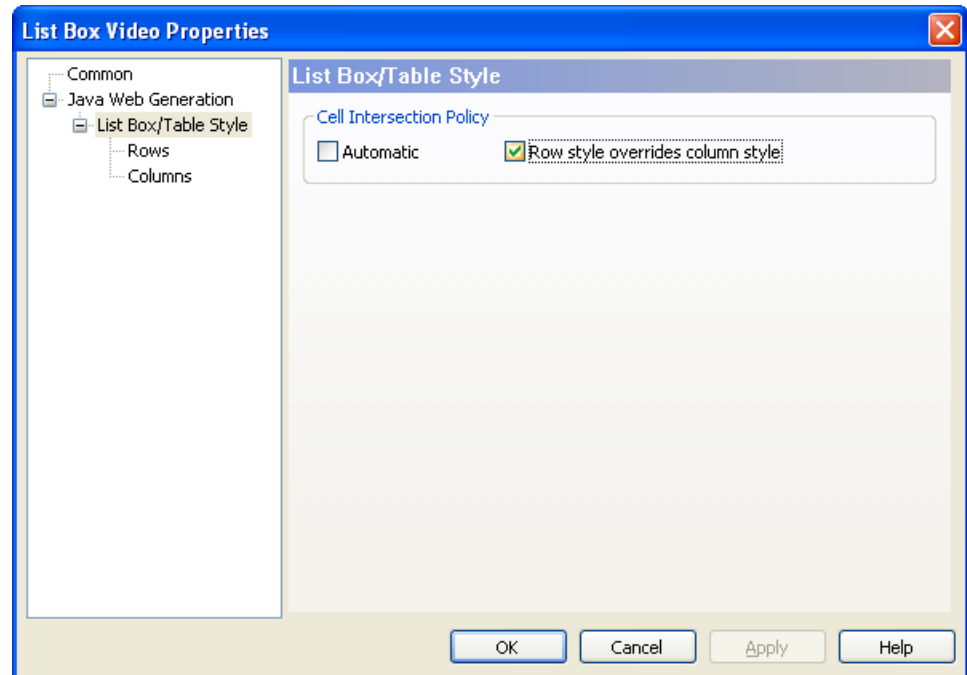
- Style schemes for all the rows and columns across the Business System
- Whether borders must display for the rows and columns

The following image illustrates the List Boxes/Tables node in the Business System Defaults Video Properties:



With the Java Web Generation node in the Navigation Diagram Video Properties dialog, you can select a custom video property to replace the default video property for specific instances of list boxes or tables. You can also specify the cell intersection policy, the rows styles and columns styles that replace the default cell intersection policy, the rows styles, and the columns styles defined in the Business System defaults video properties dialog.

The following image illustrates the List Box/Table Style node in the Navigation Diagram Video Properties:



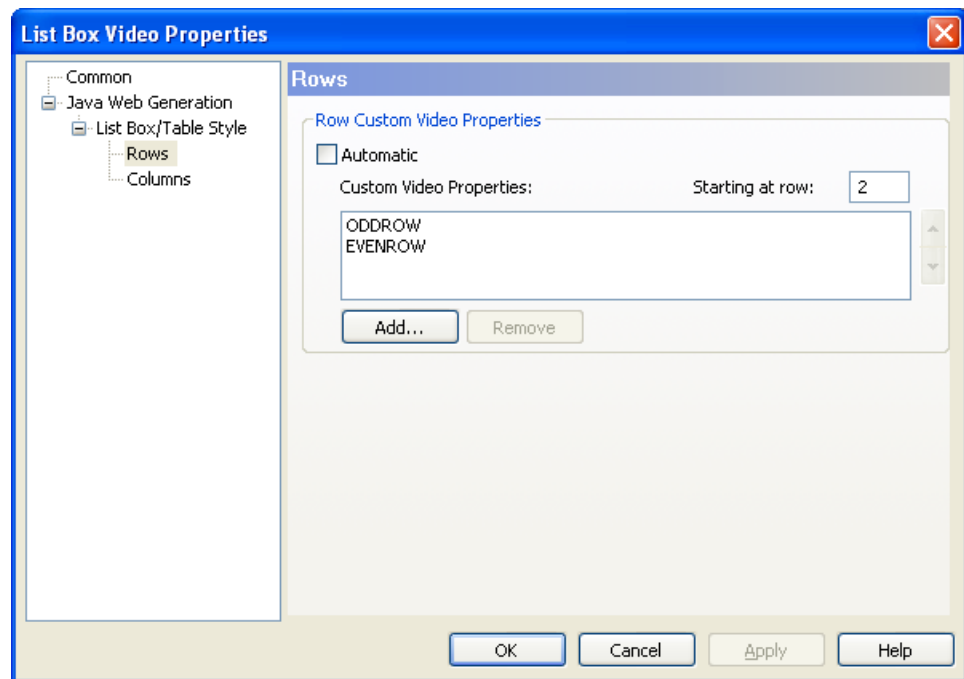
Note: For more information about specifying custom video properties for the list boxes and tables in Java Web Generation applications, see the *Toolset Help*. For more information about specifying custom video properties for rows, and columns, see the sections, Custom Video Properties for Rows, and Custom Video Properties for Columns.

Custom Video Properties for Rows

You can specify the appearance settings and style schemes for all the rows in list boxes in a Java Web Generation application by assigning custom video properties to the rows from the Business System Default Video Properties dialog. At the Business System level, you can also specify multiple custom video properties that alternate for the rows in a repeated order.

With the Rows node in the Navigation Diagram Video Properties dialog, you can select a set of custom video properties for rows to replace the default video property for the rows in specific instances of list boxes.

The following illustration shows the Rows node in the Navigation Diagram Video Properties:



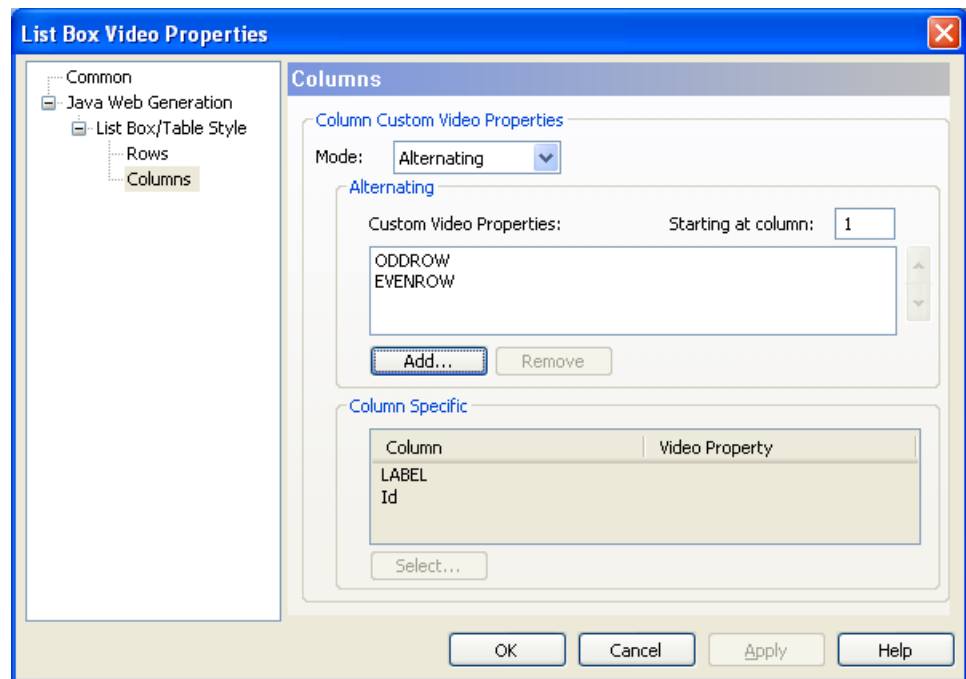
Custom Video Properties for Columns

You can specify the appearance settings and style schemes for all the columns in list boxes in a Java Web Generation application by assigning custom video properties to the columns from the Business System Default Video Properties dialog. At the Business System level, you can also specify multiple custom video properties that alternate for the rows in a repeated order.

With the Columns node in the Navigation Diagram Video Properties dialog, you can select a set of custom video properties to replace the default video property for the columns in specific instances of list boxes. With this dialog, you can also select the automatic, alternating, or column specific style schemes for the columns.

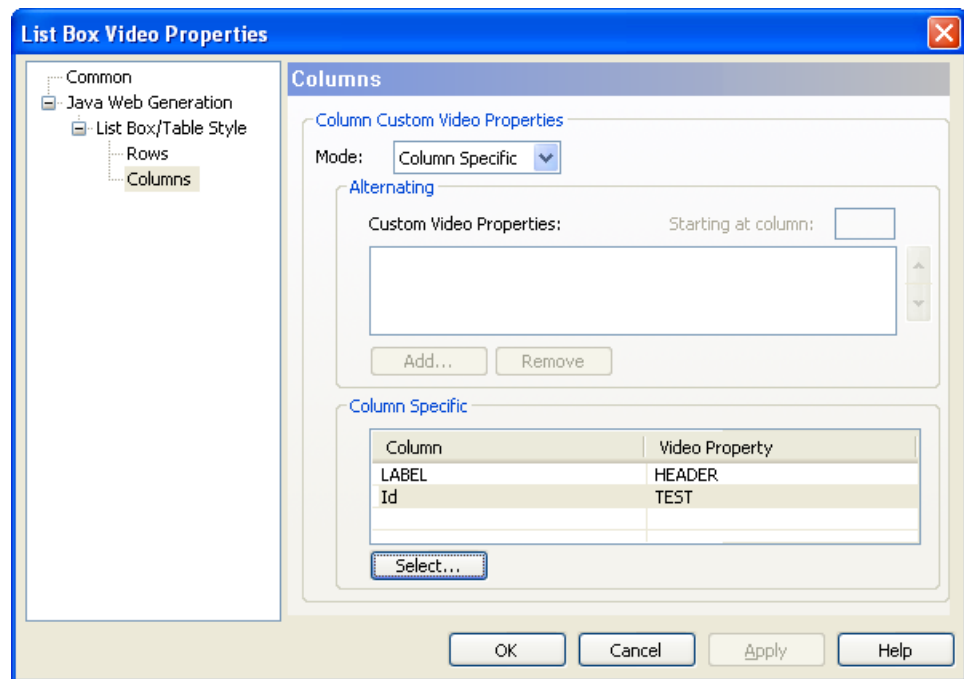
- In the automatic mode, you can apply the Business System Default Video properties setting for columns in a specific list box.
- In the alternating columns style scheme, you can specify multiple video properties that are alternated in cyclic order for the columns.
- In the column specific style scheme, you can specify a different custom video property for each list box column.

The following illustration shows the Columns node in the Navigation Diagram Video Properties with the alternating columns style scheme:



Note: If a list box or a table has only one column, you may specify only one alternating custom video properties. If you specify more than one alternating custom video property, only the first custom video property is used.

The following illustration shows the Columns node in the Navigation Diagram Video Properties with the column specific style scheme:



Cascading Style Sheets

In CA Gen, all the windows in a given Business System inherit up to two cascading style sheets from the HTML\css directory.

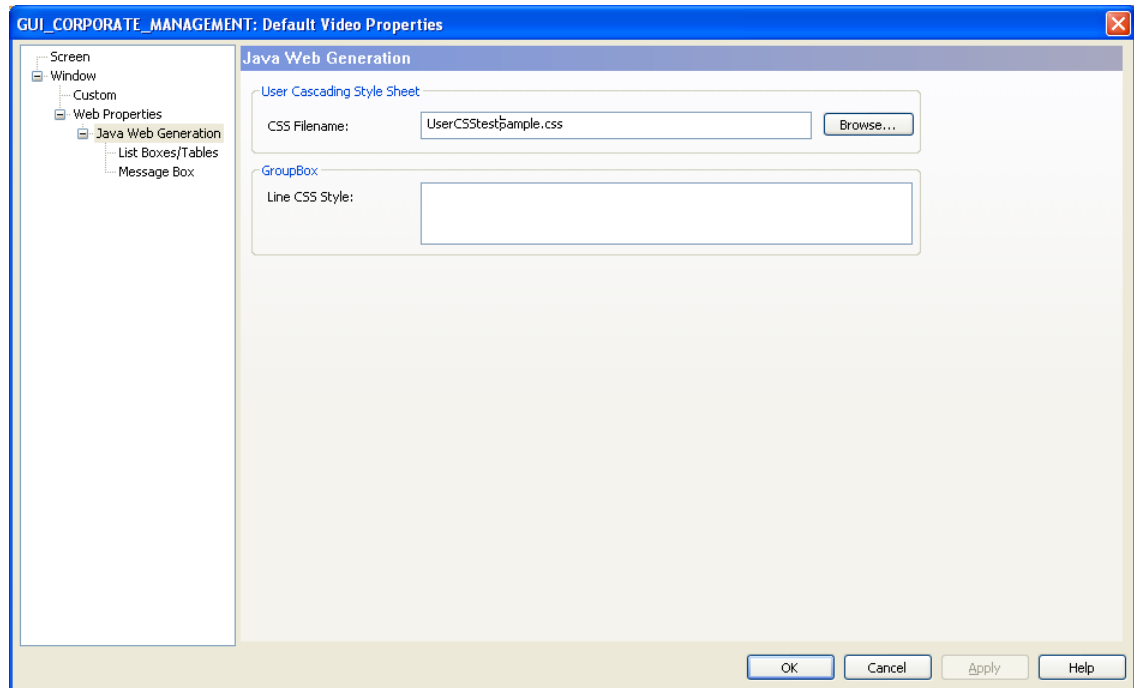
- The first stylesheet is a generated file with a name of *<Business_System_Name>.css* file. This file contains all the class selectors with their foreground color, background color, and font information specified in the Business System Default Video Properties dialog. The Class selector style statements in this file include the information from window default video properties and from custom video properties. The first link statement in each generated *<window_name>.html* file references this CSS file.
- The second stylesheet is an optional cascading style sheet file specified and supplied by the user. You can specify the css file name in the CSS Filename entry field from the Java Web Generation node in the Default Video Properties dialog. The second link statement in each generated *<window_name>.html* file references this CSS file, if this file is specified. The user must maintain, modify or validate this stylesheet file, if required. A sample user CSS file is available in the Gen\samples\CSS directory.

Note: For more information about specifying the user cascading style sheet, see the *Toolset Help*.

Line CSS Style for GroupBox Lines

You can specify the line CSS style strings for all group boxes across a Java Web Generation application. You can use the Java Web Generation node in the Business System Defaults Video Properties dialog to specify the line CSS style strings for the group boxes.

The following image illustrates the Java Web Generation node in the Default Video Properties dialog:

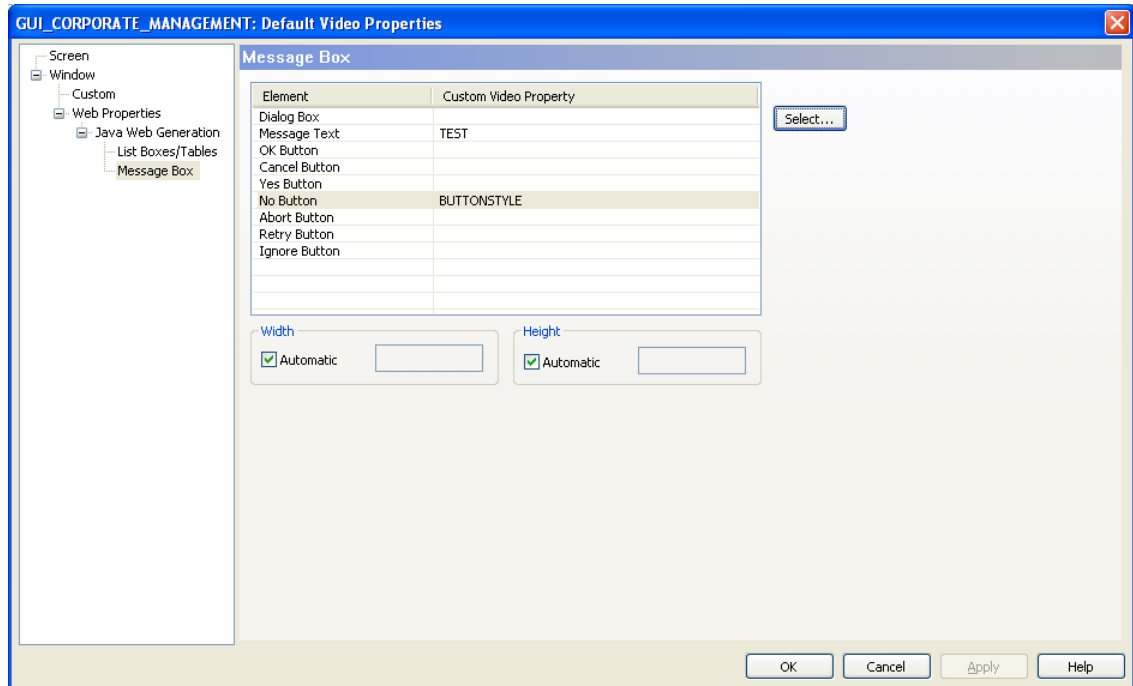


Note: For more information about line CSS style strings for group boxes, see the *Toolset Help*.

Custom Video Properties for Message Boxes

You can specify the appearance settings for message boxes in a Java Web Generation application by using the Message Box node of the Java Web Generation Node in the Business System Video Properties dialog.

The following illustration shows the Message Box node in the Business System Default Video Properties dialog:



Note: For more information about assigning Custom Video Property for Message Box, see the *Toolset Help*.

Styles of Drop down Menus

Web Generation menus are available in two styles: vertical and horizontal. In the default style for HTML menus, vertical, the menus are vertically expanded on the left side of a window. Drop down (horizontal) menus are similar to the Windows style of drop down menus. The option to switch between both styles is available on the level of the Business System. The default is to generate vertical menus.

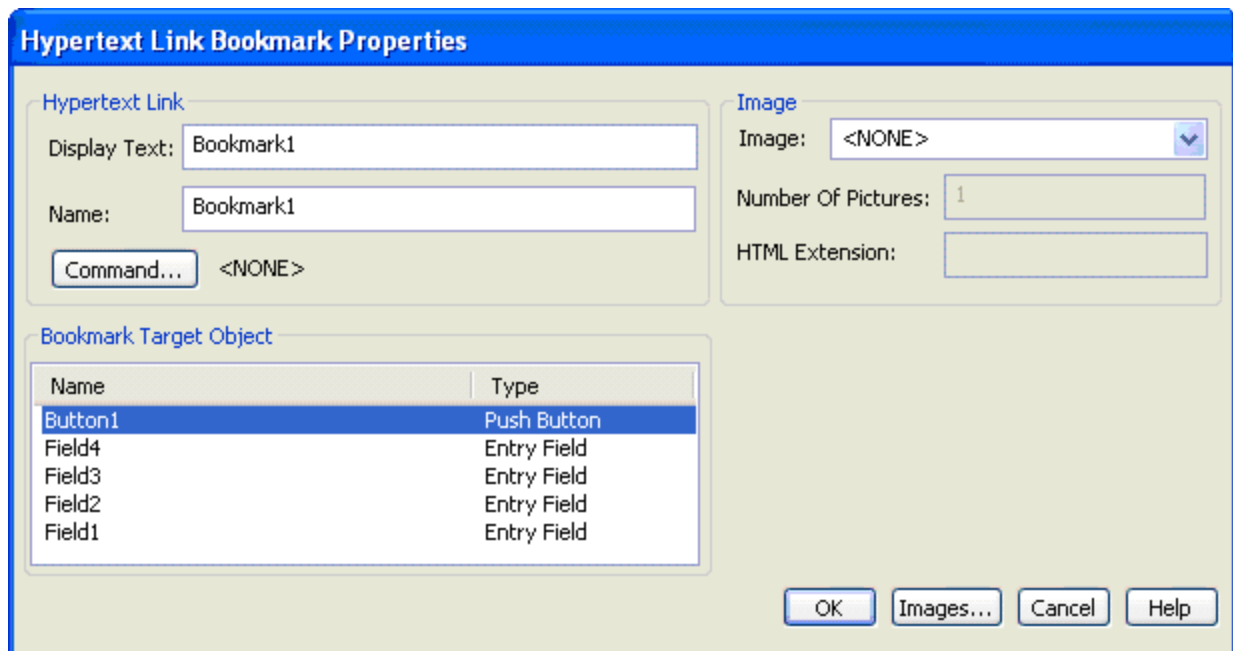
Types of Hypertext Links

CA Gen Web Generation provides two types of hypertext links to an application. They are:

- **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 options are only available in HTML mode and grayed out 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:



The dialog box is titled "Hypertext Link Bookmark Properties". It contains two main sections: "Hypertext Link" and "Bookmark Target Object".

Hypertext Link section:

- Display Text:** A text field containing "Bookmark1".
- Name:** A text field containing "Bookmark1".
- Command:** A button labeled "Command..." followed by a dropdown menu showing "<NONE>".
- Image:** A dropdown menu showing "<NONE>".
- Number Of Pictures:** A text field containing "1".
- HTML Extension:** An empty text field.

Bookmark Target Object section:

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

At the bottom right of the dialog are four buttons: "OK", "Images...", "Cancel", and "Help".

The Display Text and Name fields are populated with default values that you can replace.

Note: 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.

The dialog box is titled "Hypertext Link Open New Browser Window Properties". It is divided into two main sections: "Hypertext Link" and "New Browser Window Properties".

Hypertext Link Section:

- Display Text:** A text field containing "Browserwindow1".
- Name:** A text field containing "Browserwindow1".
- Command:** A button labeled "Command..." next to a text field containing "<NONE>".
- Image:** A section containing:
 - Image:** A dropdown menu showing "<NONE>".
 - Number Of Pictures:** A text field containing "1".
 - HTML Extension:** An empty text field.

New Browser Window Properties Section:

- URL:** A text field containing "http://www.ca.com|".
- Window Properties:** A row of checkboxes: "Menubar" (unchecked), "Toolbar" (unchecked), "Statusbar" (unchecked), "Scrollbars" (unchecked), "Resizable" (checked), and "Location" (unchecked).
- Dimensions:** Four text fields for "Top" (100), "Left" (100), "Width" (250), and "Height" (250).

Buttons: At the bottom right, there are four buttons: "OK", "Images...", "Cancel", and "Help".

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 only. 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:

Mnemonic Key:

Name:

☐ Is the Default Push Button

☒ Display As Hypertext Link

Bitmap

Bitmap:

Number of Pictures:

☒ Adjust Button to Bitmap size

HTML Extension:

Accelerator

☐ Ctrl ☐ Alt ☐ Shift Key:

Button Action

☐ Special Action

☐ Initiates the Dialog Box

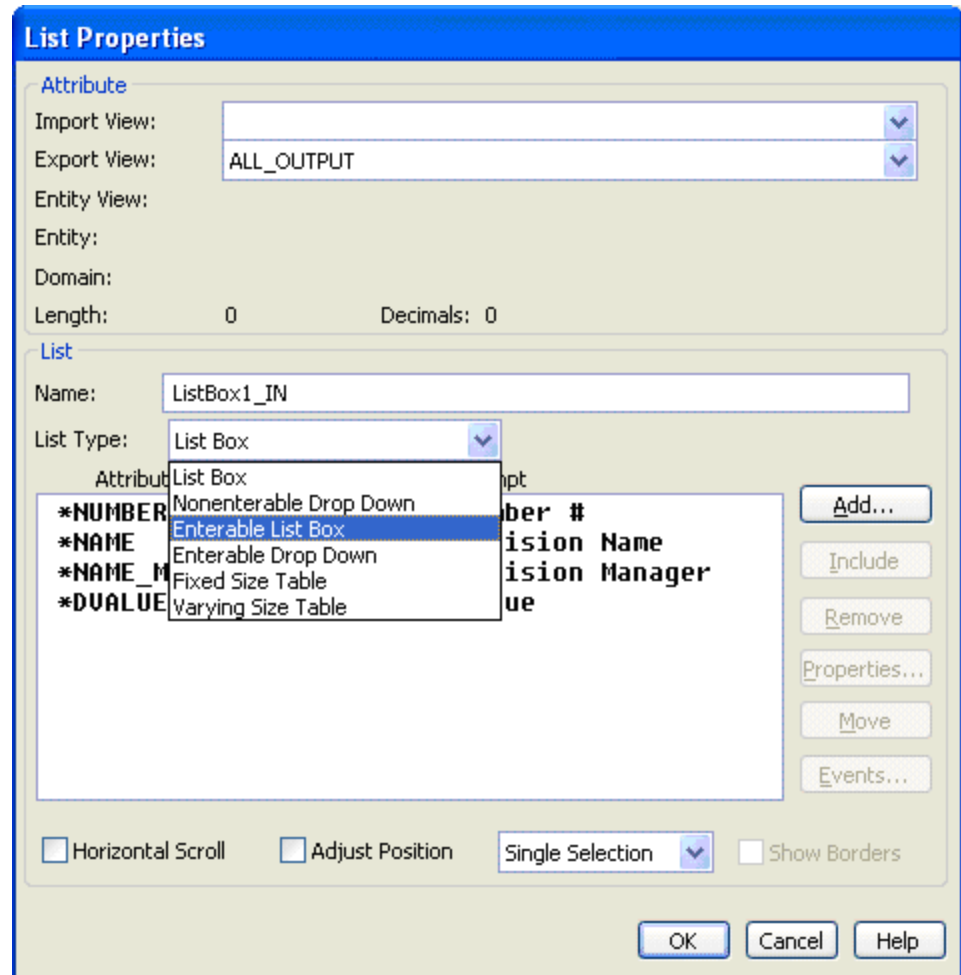
☒ Executes the Procedure Step

Varying and Fixed Size Tables

HTML Tables are a type of list that can be associated with group views. CA Gen Web Generation 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 illustration details the List properties dialog:



Note: If Fixed Size Table or Varying Size Table is selected as List Type, the Horizontal Scroll check box is enabled. When this check box is selected, the table does not have a horizontal scroll bar but the table is allowed to grow beyond the designed width at runtime, provided that the designed table width is smaller than the total width of all columns.

CA Gen not only enables users to design dynamic and fixed size tables, but also to import these two kinds of tables from external HTML.

Change Multi-State Images Dynamically

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. Web generation provides a similar capability. Where for GUI applications you had a different section of the provided bitmap designated for each state, with Web Generation, 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 must 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.

Embed ActiveX Controls

Web Generation supports embedding ActiveX controls and defining their event management in a similar way that ActiveX controls are currently designed in GUI clients.

When designing OCX controls for Web Generation, the following performance guidelines need to be considered:

- Any statement that Gets or Sets a property or invokes a method on the ActiveX Control requires the runtime to return to the browser.
- Any statement that does a Boolean compare of two OCX GUI Objects or objects returned from Get or Invoke statements requires the runtime to return to the browser.
- Excessive Action Block manipulation of an OCX control causes performance degradation, especially for an Internet application.

Restrictions on the Use of OCX Controls

OCX controls rely heavily on Microsoft technologies and can only execute in browsers running on Windows platforms. OCX controls can be rendered using Internet Explorer without the need to install any third party software. Mozilla Firefox does not natively support OCX controls. CA supports OCX controls on Internet Explorer only. OCX controls can be executed on any supported Application Server on a supported J2EE platform.

For the OCX controls to be displayed on a browser machine that does not have OCX controls installed, License Package File (LPK) needs to be generated. The LPK file contains runtime licenses for the ActiveX controls.

To generate the LPK, and include the genocx.lpk in the WAR file

1. Generate License Package File (LPK):
 - a. Download LPKTool.exe from <http://www.microsoft.com/downloads/details.aspx?familyid=d2728e89-575e-42e9-a6ff-07d0021e68cc&displaylang=en> to the machine that has licensed ActiveX controls installed.
 - b. Run LPKTool.exe. Highlight each licensed ActiveX control that will be used in the application and click Add.
 - c. Click Save and then Exit. Type the name **genocx.lpk**.
2. Include genocx.lpk in the WAR file:
 - a. During Assembly from the build tool, in the EAR File Assemble Details dialog, click Additional Files tab.
 - b. Select <WAR file>.
 - c. Click Include File and include genocx.lpk along with the OCX files that will be used in the application.
 - d. Click OK and complete the deployment process.

PAD control of OCX components is limited to the current Procedure Step. That is, the GUI object representing the component cannot be passed via the views to another Procedure Step manipulation.

Web Generation does not support features that cannot be modified through methods and properties of an OCX control. This may include features that are currently available to GUI clients by MFC programming. An example is specifying Sequencing of Tabbing among controls that are embedded on OCX Tab controls.


In Web Generation Applications, message boxes and OCX controls require threading to be enabled in the Assemble utility of the Build Tool. However, some Application Servers, including WebSphere may not work correctly if threads are used.

Action Block statements that manipulate ActiveX Controls cannot be placed in an Open Event or in the Execute First section of the Action Block. This restriction is there because at the time of Action Block execution, the html page containing the control has not yet been returned to the browser. Because of this restriction, it is recommended that the designer use the Load concept, where after the html page is loaded, the end user interacts with the page and initializes the control.

ActiveX Controls are only valid for a single window execution context. That is, after a window replaces the current window (the window with the ActiveX 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.

Browser Prototype Mode

This feature is specific to Edit Mode HTML only. It is available in the Navigation Diagram to allow the user to invoke the default browser and display HTML. It gives the developer a preview of the interface for the selected window or dialog. In the Navigation Diagram, select Prototype from the Mode menu, and click the Browser.

Note: An icon  is also available from the Navigation Diagram toolbar to access the Browser Prototype Mode.

Consistency Check Rules

Consistency check rules identify the unsupported features in Web Generation applications. These warnings display when unsupported features are used in any packaged procedure step whose load module could potentially be generated for Web Generation, whether it is currently being generated or not. For a complete set of unsupported features, please refer to the corresponding chapter in this guide.

Design Considerations

When creating the user interface for your client application, please note that the Toolset representation will not be identical to what you see at runtime. There will be some 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 can expect.

Note that the following items are by no means a comprehensive list of the differences between GUI, ASP.NET Web Clients, and Web Generation.

- 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 drop down list—In Web Generation, a drop down list, used to display permitted values, displays zero (0) at runtime for a numeric field, if zero is one of the permitted values, regardless of whether a user specifies a default value or not.

For a text field, spaces display if space is one of the permitted values. If zero or space is not one of the permitted values, the default displays. If there is no default, and zero (or space) is not a permitted value, <NONE> displays. The <NONE> value is not a valid selection. Selecting <NONE>, or making no selection sends an input of zero (0) or space to the application, depending on the domain of the field

A drop down list for permitted values displays the prompt, and does not display the actual value. A list box defined as a non-enterable drop down displays <NONE> as the first line in the list. However, <NONE> is not a valid selection. Selecting <NONE>, or making no selection, does not return a valid row selection to the application. When you do GET ROW HIGHLIGHTED ...GIVING SUBSCRIPT ..., the subscript is zero (0) and any attempt to reference the row thus selected may cause a Java runtime exception. The value of the default display of <NONE> can be modified by changing *ief_error.js*, but you must define a value for the variable NONE.

- Fixed and Varying Size Tables—Fixed and Varying size tables are supported in HTML Mode and ASP.NET Mode. If you create a Table in HTML Mode with Common Edit Modifications ON and then switch to Window mode, the table type will change to a list box.
- Primary Windows and Primary Dialogs are always displayed as a full internet browser page. Secondary Dialogs can be displayed either as a full internet browser page or as pop-up dialogs; similar to the way they are displayed in GUI applications. The option to use pop-up dialogs can be selected while assembling the application.
- In Web Generation, due to performance reasons, text entered into Entry Fields is validated and uppercased only when they lose focus.

Web Generation applications allow opening more than one dialog box at a time. There are differences between how GUI and Web Generation applications handle multiple secondary dialog boxes.

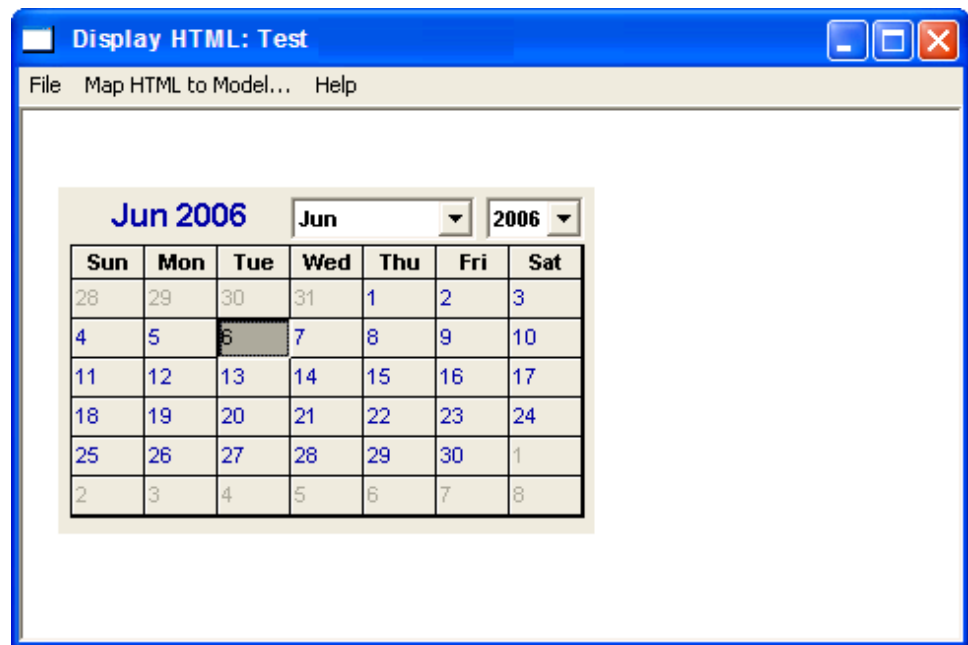
In Web Generation applications, following are some of the differences you can expect:

- Suppose that there is more than one dialog box open, and a link flow is initiated. Upon its return, only the dialog box that had focus and its primary window will be reopened. Secondary dialog boxes which did not have focus will not be reopened.

- In browser environments, modality is relative to the primary window only. However this modality may be subsequently impacted if multiple modal dialogs or message boxes are opened at the same time.
- The same rules listed above for secondary dialog boxes apply to message boxes.

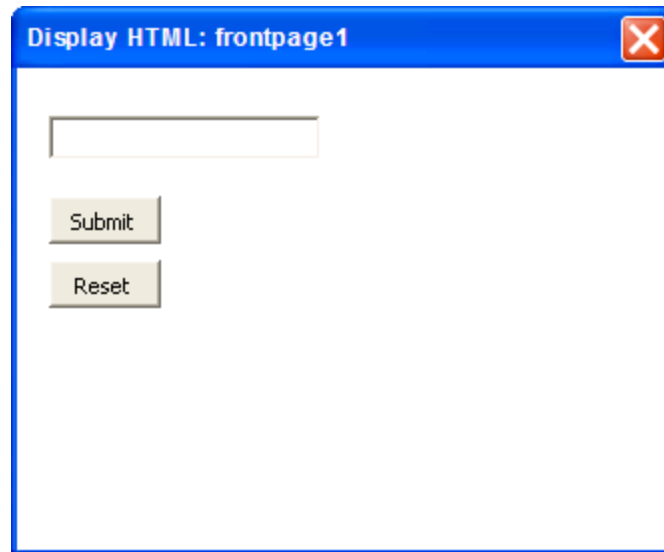
Map ActiveX Controls Using HTML Editing

This feature enables you to import and map ActiveX Controls. The same rules apply to imported ActiveX controls as to those designed in the model.



Map Submit and Reset Buttons

Submit and Reset buttons are very frequently used in HTML files. You can map Submit and Reset buttons created by third-party web authoring tools to CA Gen push buttons. The HTTP service method supported on submitting is GET, not POST.



Administration

After the externally created or edited HTML file has been imported into the model, the generator no longer generates `window_name.html` for that window. Your modified HTML file is used from that point onwards. You are responsible for storing and version controlling these files external to CA Gen in the same way you administer bitmaps. However, the Build Tool packages these files in the appropriate manner that ensures their proper assembling by the Build Tool on the target Application Server. Also, you are responsible for maintaining the `<UserCascadingStyleSheet.css>` file in the `html\css` directory and the image files in `html\images` directory of the model.

Web Generation Runtime Features

This section describes different web generation runtime features.

Using HTML Pages without Frames

CA Gen dynamically creates a frameset at runtime. Controls on any given window or dialog are included in a single frame. Any JavaScript that needs to execute in conjunction with any given page is generated in a separate JavaScript file.

This generation style includes the following benefits:

- A reduced number of generated HTML files for each window.
- Simplified generated code.
- Enhanced ability of third party Web authoring tools to read CA Gen-generated pages.

Browser Event Queue

A queue controls the processing of Browser Events. All events from the queue are processed one at a time in the order they are received. However, if multiple events of the same event type are triggered from the same control, only the first one is queued. All the following events are ignored. If multiple events of different event types are triggered from the same control, or if multiple events are triggered from different controls, they are all queued for processing.

Support for ScrollTop and ScrollBottom Events

CA Gen supports ScrollTop and ScrollBottom events for the list box. When either of these events is associated with a list box, the action block can perform event handling such as repopulating the list box with a new set of data. ScrollTop and ScrollBottom events occur when the displayed rows reach the top or bottom of the range.

Applet-Free Approach

CA Gen uses Dynamic HTML and JavaScript to perform functionality such as permitted value and edit pattern validation. Web Generation does not require the use of applets for any of its operations. Thus, enabling applet support in browsers is not required.

Enhanced List Box Scrolling and Extended Selection

Standard HTML Tables do not have scroll bars like list boxes in GUI applications. CA Gen has the ability to create a custom control, list box, which has scroll bars with a scrolling thumb associated with HTML tables. This approach complies with the GUI feel of the applications you are porting to the Web.

The scrolling thumb appears whenever the display area happens to be too small (vertically or horizontally) for displaying the available number of rows or columns. The scrolling thumb's length corresponds to the percentage of the rows displayed (the length of the thumb is inversely proportional to the length of the text). The thumb's position determines which part of the text is being viewed.

You can click the scrolling thumb and slide the mouse upward or downward (or left or right for horizontal scrolling) to cause the text in the list box to move accordingly.

The list boxes in CA Gen provide single and multiple selection capabilities. For multiple selection list boxes, extended selection is also available. Extended selection lets you select a row, then press the Shift or Control keys and select other rows. This feature is only available for list boxes on Internet Explorer. Extended selection is not supported on Mozilla Firefox or for drop down lists, Enterable list boxes, or Varying or Fixed Size Tables.

Disabled By

Many GUI applications disable controls depending on the values or states of other controls on the same page. A Disabled control appears grayed out and a user of the application cannot select it. Disabled By rules, which specify conditions under which a control is disabled, is supported by Web Generation. This facilitates porting GUI models to the Web.

Cookie-Free Approach

CA Gen Web Clients do not require cookies. Standard J2EE session management techniques are implemented to manage session and state data. The Cross-Context Flows feature uses a novel approach to managing sessions when communicating across separately installed components.

DBCS Character Entry

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.

SetBitmapName to Specify Images Dynamically

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 web applications. By using Dot Notation in the Action Diagram and invoking `SetBitmapName` you can set or modify a bitmap on its associated picture control. The bitmap pathname for Web is the same standard as the filesystem/UNC path as GUI. The Internet type path (HTTP URL) is not supported.

Accessibility

Support for Accessibility features enables people with disabilities to use Web Generation applications. Several regulations are in place worldwide to make sure that users with disabilities can use Web Applications.

Section 508 is a U.S regulation that requires Federal Agencies' Electronic and Information Technology be accessible to people with disabilities. Disability Discrimination Act (DDA) is the U.K. government's mandate containing guidelines with which software must comply so that people with disabilities can access the software.

Users will be able to navigate through the applications solely using the keyboard without needing to use the mouse. Tab key can be used to move from one control to another, arrow keys to navigate within a control and spacebar to select or click a control. Mnemonics are supported on push buttons and menus. Screen Readers like IBM Home Page Reader can also be used to access Web Generation applications.

List Box

The following table describes the keys and their functions for the list box:

Key	Function
Tab	To navigate from one cell to the other
Up Arrow	To move focus to the cell above
Down Arrow	To move focus to the cell below
Left Arrow	To move focus to the cell on the left
Right Arrow	To move focus to the cell on the right
Page Up	To move up contents of the list box by one page
Page Down	To move down contents of the list box by one page
Home	To move focus to the first column of the current row
End	To move focus to the last column of the current row
Ctrl+Home*	To display the contents of the list box starting from the first row
Ctrl+End**	To display the contents of the list box showing the last row
Spacebar	To select a row, click Event
Spacebar twice	To select a row, double click Event
Ctrl+Right Arrow***	To move focus to the next control in the tab sequence
Shift+Ctrl+Right Arrow***	To move focus to the previous control in the tab sequence

Fixed and Varying Size Tables

The following table describes the keys and their functions for the tables:

Key	Function
Tab	To navigate from one cell to the other
Home	To move focus to the first column of the current row
End	To move focus to the last column of the current row
Spacebar	To select a row, click Event
Spacebar twice	To select a row, double click Event
Ctrl+Right Arrow***	To move focus to the next control in the tab sequence
Shift+Ctrl+Right Arrow***	To move focus to the previous control in the tab sequence

Note:

- * Instead of Home key, users can use any key except TAB key and modifier keys if they specify it in the ief_Make.js. In CA/Gen/ief_Make.js, LBHomekey is set to 36/Home key ("var LBHomekey = 36;") by default. But it can be changed to the keycode they wish to use except 9 (tab key), 16 (shift key), 17 (ctrl key) and 18 (alt key).
- ** Instead of End key, users can use any key except TAB key and modifier keys if they specify it in the ief_Make.js. In CA/Gen/ief_Make.js, LBHomekey is set to 35/End key ("var LBHomekey = 35;") by default. But it can be changed to the keycode except 9 (tab key), 16 (shift key), 17 (ctrl key) and 18 (alt key).
- *** Instead of Right Arrow key, users can use any key except TAB key and modifier keys if they specify it in the ief_Make.js. In CA/Gen/ief_Make.js, LBMovefocus is set to 39/Right Arrow ("var LBMovefocus = 39;") by default. But it can be changed to the keycode except 9 (tab key), 16 (shift key), 17 (ctrl key) and 18 (alt key).

Vertical Menus

The following table describes the keys and their functions for the vertical menus:

Key	Function
Tab	To navigate from one menu item to the other
Down Arrow	To move focus to the menu item below
Up Arrow	To move focus to the menu item above

Key	Function
Right Arrow	To move focus to the first item in the sub menu
Left Arrow	To move focus from sub menu to parent menu
Escape	To close the sub menu and move focus to the parent menu

Horizontal Menus

The following table describes the keys and their functions for the horizontal menus:

Key	Function
Tab	To navigate from one menu item to the other
Down Arrow	To move focus to the sub menu
Up Arrow	To move focus to the menu item above
Right Arrow	To move focus to the next menu item
Left Arrow	To move focus to the previous menu item
Escape	To close the sub menu and move focus to the parent menu

Radio Buttons

The following table describes the keys and their functions for the radio buttons:

Key	Function
Tab	To move focus to the radio button group
Down Arrow	To select the next item in the group
Up Arrow	To select the previous item in the group

Check Boxes

The following table describes the keys and their functions for the check boxes:

Key	Function
Tab	To move focus to the check box
Spacebar	To select the check box

Drop down List

The following table describes the keys and their functions for the drop down list:

Key	Function
Tab	To move focus to the drop down list
Down Arrow	To select the item below the current item
Up Arrow	To select the item above the current item

Hypertext Links

The following table describes the keys and their functions for the hypertext links:

Key	Function
Tab	To move focus to the hypertext link
Enter Key	To click the hypertext link

Web Generation Additional Features

Additional options that affect the look and feel of your application are available for you to customize during assembling.

Note: For more information, see the *Build Tool User Guide*.

Chapter 5: Generating a Web Generation Application

After designing your Web Generation components, you are ready to generate your application. You can use either the Toolset or the Client/Server Encyclopedia for generation. A load module is considered eligible for Web Generation if the Web Generation settings are reflected in any one of the following:

- The Generation Defaults dialog (when the Override Bus Sys Target Environment with above defaults option is checked).
- The Client Environment.
- The Business System Environment.

Regeneration of Web Generation Applications

Due to the need to change the version number portion of the package names used in generated code and runtime between releases, you must regenerate your Web Generation applications when upgrading to any release of CA Gen.

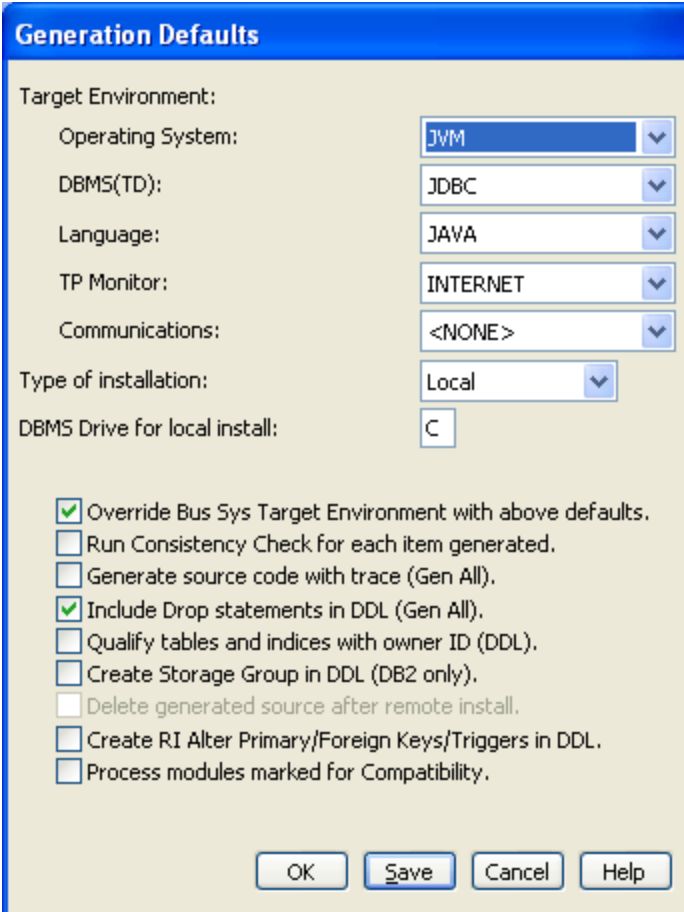
Generate a Web Generation Application from the Toolset

After the Web Generation installation and design procedures are completed, you can generate your Web Generation application from the Toolset.

To generate a Web Generation application from the Toolset

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

- From the Generation Window, select the Options Menu and then Generation Defaults. The Generation Defaults dialog appears.



The image shows a dialog box titled "Generation Defaults". It contains several configuration options:

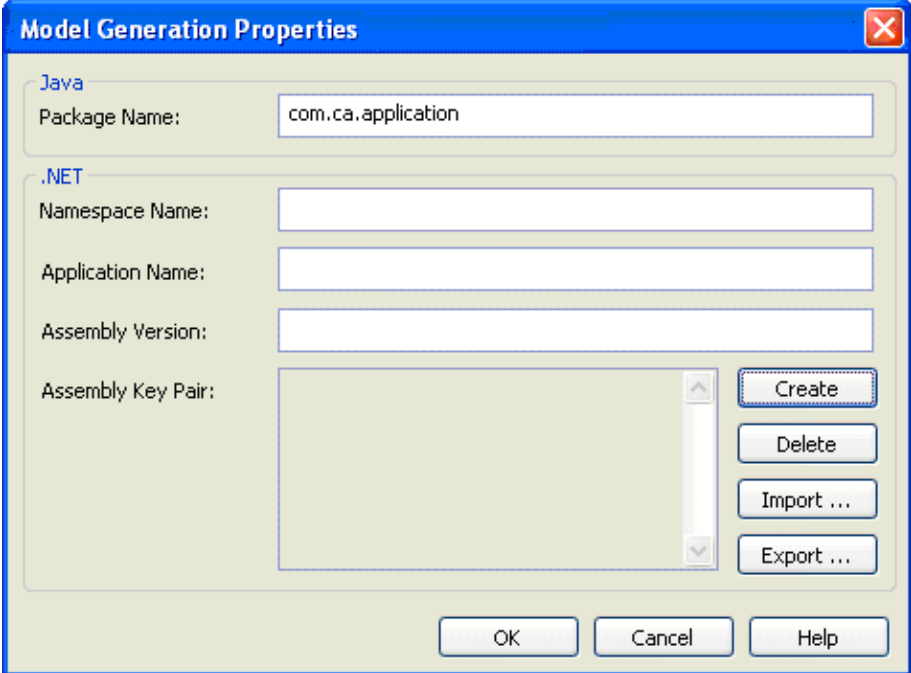
- Target Environment:**
 - Operating System: JVM (selected in a dropdown)
 - DBMS(TD): JDBC (selected in a dropdown)
 - Language: JAVA (selected in a dropdown)
 - TP Monitor: INTERNET (selected in a dropdown)
 - Communications: <NONE> (selected in a dropdown)
- Type of installation:** Local (selected in a dropdown)
- DBMS Drive for local install:** C (text input)
- Checkboxes:**
 - ☒ Override Bus Sys Target Environment with above defaults.
 - ☐ Run Consistency Check for each item generated.
 - ☐ Generate source code with trace (Gen All).
 - ☒ 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.

At the bottom are four buttons: OK, Save, Cancel, and Help.

- At the Operating System drop-down box, select JVM.
- At the Language drop-down box, select Java.
- At the Type of Installation drop-down box, select either 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 copying onto a server running on the same machine as the toolset. The default is Local.
- Click OK.

- (Optional) You can select the Options Menu from the Generation Window, then the Model Generation Properties and enter a name in the Java Package Name box.

The image shows a 'Model Generation Properties' dialog box with a blue title bar and a close button (X) in the top right corner. The dialog is divided into two sections: 'Java' and '.NET'. The 'Java' section has a 'Package Name' text box containing 'com.ca.application'. The '.NET' section has four text boxes: 'Namespace Name', 'Application Name', and 'Assembly Version', all of which are empty. Below these is a list box for 'Assembly Key Pair' which is also empty. To the right of the list box are four buttons: 'Create', 'Delete', 'Import ...', and 'Export ...'. At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

The Java Package Name is used for all generated Java files.

- Click OK and you are ready to continue with Generation, Build, Assembling, and deployment of a Web Generation Application.

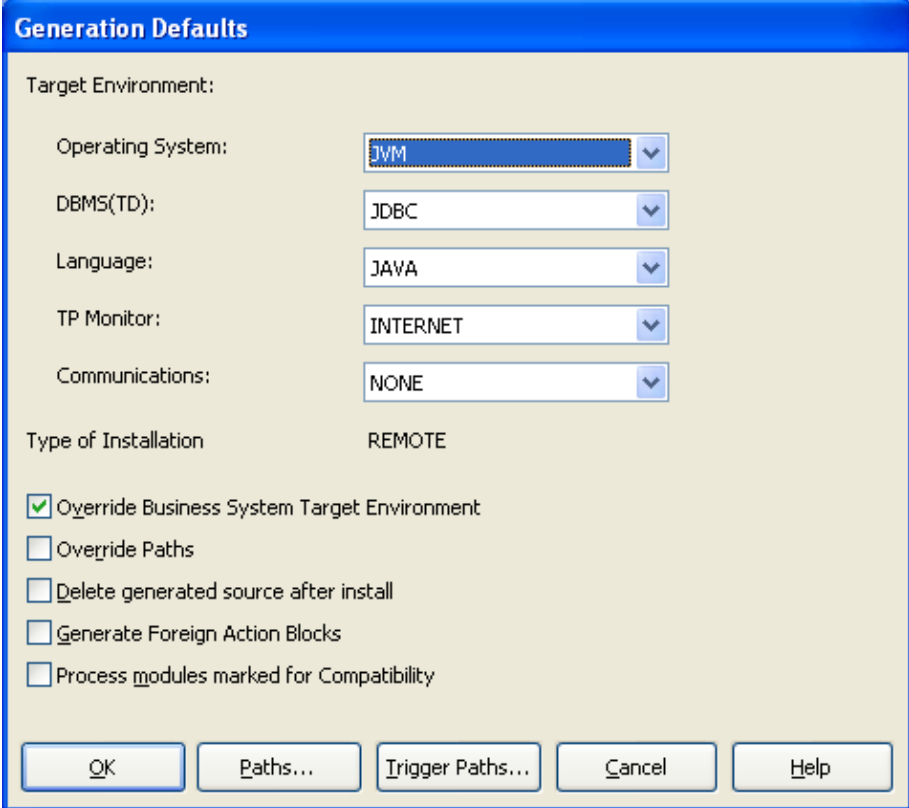
Generate a Web Generation Application from CSE

You can generate Web-enabled applications from the Client/Server Encyclopedia (CSE).

To generate a Web Generation Application from CSE

- Start the Construction Client and open your model.
- Select Code, Generate, Cooperative.

3. From Cooperative Application Generation, select Defaults and the following Generation Defaults dialog displays.



The image shows a 'Generation Defaults' dialog box with a blue title bar. It contains several configuration options for a target environment. The 'Target Environment' section includes five dropdown menus: 'Operating System' (set to JVM), 'DBMS(TD)' (set to JDBC), 'Language' (set to JAVA), 'TP Monitor' (set to INTERNET), and 'Communications' (set to NONE). Below these is a 'Type of Installation' field set to 'REMOTE'. A group of five checkboxes follows: 'Override Business System Target Environment' (checked), 'Override Paths' (unchecked), 'Delete generated source after install' (unchecked), 'Generate Foreign Action Blocks' (unchecked), and 'Process modules marked for Compatibility' (unchecked). At the bottom are five buttons: 'OK', 'Paths...', 'Trigger Paths...', 'Cancel', and 'Help'.

Generation Defaults	
Target Environment:	
Operating System:	JVM
DBMS(TD):	JDBC
Language:	JAVA
TP Monitor:	INTERNET
Communications:	NONE
Type of Installation	REMOTE
<input checked="" type="checkbox"/> Override Business System Target Environment	
<input type="checkbox"/> Override Paths	
<input type="checkbox"/> Delete generated source after install	
<input type="checkbox"/> Generate Foreign Action Blocks	
<input type="checkbox"/> Process modules marked for Compatibility	
OK Paths... Trigger Paths... Cancel Help	

4. From the Operating System drop-down list, select JVM.
5. From the Language drop-down list, select Java.

After defining the Web Generation components, you are ready to continue with, generation, build, assembling then Deployment of a Web Generation Application.

Chapter 6: Building and Running a Web Generation Application

Before assembling the Web Generation application, you should convert bitmap images (.bmp files) to a different format such as .jpeg, .jpg or .gif. You need to convert the image files using a third-party tool 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 Window/Dialog Properties dialog from the Navigation Diagram in the Toolset to match the file extension for the format you are using.

Note: For a discussion on using External Action Blocks, see the *Enterprise Java Bean User Guide*.

After generating your Web Generation application, three new directories are created under the model's local directory to accommodate the Web Generation components. These new directories are:

- Java directory (contains the generated Java source file)
- HTML (contains HTML, JSP, CSS, and JS files)
- HTML\Images directory (contains converted images such as JPG, GIF, and PNG)

Build and Assemble Web Generation 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.

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

Configure Web Generation Communications

The Web Generation 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.properties` file. `commcfg.properties` 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 EAR file.

Note: For more information about the `commcfg.properties` file, see the *Distributed Processing - Overview Guide*.

Content Compression

Compressing the content that is transferred across the network can improve performance of your applications. The fewer the number of bytes of data that is sent over the network, the faster the transfer will occur.

The following two types of content that can be compressed by Web Generation:

- **Static Content** such as html, js, and css files may be compressed prior to deployment of the application.
- **Dynamic Content** is content that is created while the application is being used and must be dynamically compressed.

Both static and dynamic compressions use the GZIP compression technique.

For static compression, compressed files are packaged in the application WAR file and deployed in their compressed form.

For dynamic compression, the content is sent through a filter and compressed on-the-fly by the CA Gen runtime.

Both Microsoft Internet Explorer and Mozilla Firefox will recognize the GZIP compression format and will decompress the content.

Static and Dynamic Compression can be enabled or disabled using the Build Tool. By default, both types of compression are enabled.

Run Web Generation

To be able to run a Web Generation application, you need to deploy it under an Application Server. Use the EAR file created by the Build Tool under your model's `java\delploy.j2ee` directory. Follow your Application Server's deployment instructions to actually deploy the EAR file.

Your generated application can be invoked from a supported browser. The URL to access Web Generation applications will rely on the context specified plus the trancode. It will follow one of the following formats:

For example: `<http://<hostname>/<context>/trancode.jsp>`
can now be accessed through
`<http://<hostname>/<context>/trancode>`.

The *web.xml* file is also used to modify the Session Timeout value, which is set to 30 minutes by default.

GUI applications allow you to define CLEAR SCREEN INPUTS at execution time, which affect the processing of the application. For Web Generation, this variable must be either set in the JSP before deployment or specified as a URL parameter. After generating the application and before assembling and deploying, you must add the following line of code to JSP, which serves as the entry point of the application:

```
request.setAttribute("clearScreenInputs", "myValue");
```

The additional code must come between the `<%` and the `%>`.

An alternative to modifying the generated .jsp file is to add the `clearScreenInputs` parameter in the URL. For example you can enter something similar to the following:

```
http://<host:port>/<context>/<trancode>.jsp?clearScreenInputs=a,b,c..
```

Support for Tabbed Browsing

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

A tabbed session is comparable to an individual browser instance of the Web Generation 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 and Firefox 2.x.

Chapter 7: HttpServletRequest and HttpServletResponse Object Access

This chapter describes the `HttpServletRequest` and `HttpServletResponse` classes and how to access them.

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

HttpServletRequest

`HttpServletRequest` is a class that represents a request sent by the browser to a servlet using the HTTP protocol. See the J2EE documentation for a list of the methods and properties of this class.

Access Method

The Java version of Web Generation provides two mechanisms for accessing the `HttpServletRequest` object: user exits or external action blocks.

Accessing through the following set of user exits under the CA Gen install directory hierarchy:

- `\exits\coopflow\ejbrmi\EJBRMIContextExit.java`
- `\exits\coopflow\ejbrmi\EJBRMIDynamicCoopFlowExit.java`
- `\exits\coopflow\ejbrmi\EJBRMISecurityExit.java`
- `\exits\coopflow\tcpip\TCPIPDynamicCoopFlowExit.java`
- `\exits\msgobj\cfb\CFBDynamicMessageDecryptionExit.java`
- `\exits\msgobj\cfb\CFBDynamicMessageEncryptionExit.java`
- `\exits\msgobj\cfb\CFBDynamicMessageSecurityExit.java`

An object named `runtimeObject` is exposed in the above set of user exits. In a web context, the `HttpServletRequest` object will be available as the `runtimeObject`. To utilize this object as an `HttpServletRequest` object, add the following to the import section of the user exit you are modifying:

```
import javax.servlet.*;
import javax.servlet.http.*;
```

Add the following code to any method that uses the object, if:

```
((runtimeObject != null) && (runtimeObject instanceof HttpServletRequest)) {  
    HttpServletRequest request = (HttpServletRequest) runtimeObject;  
    /**  
    * User-written code should be inserted here  
    */  
}
```

Accessing through External Action Blocks:

To use the HttpServletRequest object

1. Design an External Action Block (EAB).
2. Use the EAB in the Pstep/Action Block where access to the requested information is desired.
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  
HttpServletRequest req = (HttpServletRequest)(iefRuntimePam2.getRequest());
```

6. Query the desired header as:

```
String user = req.getRemoteUser();
```
7. Set system attributes as:

```
globdata.getStateData().setUserId(user);
```
8. Set attributes in views as:

```
w_oa.exservletreq_servletrequest_scheme.set(req.getProtocol());
```
9. Use the values set in the views of the application logic.

Note: CA Gen global data (globdata) is available here but CA strongly recommends that users do not use it, as it is subject to change.

10. Compile the application.

HttpServletResponse

HttpServletResponse is a class that contains HTTP protocol-specific data. It is usually sent to the browser using the HTTP protocol. See the J2EE documentation for a list of the methods and properties of this class.

Accessing methods of the `HttpServletResponse` class in Web Generation applications is risky and could potentially cause runtime errors. For example, invoking the methods `sendRedirect` or `sendError` circumvents essential execution paths in the Gen runtime that impact history handling. Accessing certain objects may also cause the servlet container to throw exceptions. These include streams or writers via `getOutputStream` or `getWriter`. Also `reset` or `resetBuffer` may cause the servlet container to throw exceptions.

Access Method

To use the `HttpServletResponse` object

1. Design an External Action Block (EAB).
2. Use the EAB in the Pstep or Action Block where access to the response is desired.
3. Design the views and attributes to 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.  
HttpServletResponse res = (HttpServletResponse)  
(iefRuntimeParm2.getResponse());
```

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

```
Cookie myCookie = new Cookie (name,value);  
res.addCookie(myCookie);
```
7. Compile the application.

Chapter 8: Messages

This chapter refers to the Consistency Check messages and Java Web Generation error messages. The Java Web Generation error messages are grouped in numeric ranges that represent certain types of errors.

Consistency Check Messages

New consistency check rules identify the unsupported features in Web Generation applications. For a list of consistency check messages, see the following CA Gen Online Help topics for the ranges listed:

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

Web Generation General Error Messages 7000-7999

Exception	Reason	Action
7000	An illegal operation occurred. This is the most general and worst type of error.	None. Contact your Web Server Administrator.
7001 - ClassNotFoundException	Server CLASSPATH is improperly configured or the class does not exist.	<ul style="list-style-type: none">■ Include the path where the class file exists in the CLASSPATH.■ Include the jar file where the class file exists in the CLASSPATH.
7002 - IOException.	I/O operation either failed or was interrupted.	Verify the operation target.
7003 - IllegalAccessException	The current executing method does not have access to the definition of the specified class because the class is not public or in another package.	Verify the packaging of the Web Generation classes.

Exception	Reason	Action
7004 - InvocationTargetException	The method for the specified class either does not exist or is inaccessible.	Verify the application class.
7005 - InstantiationException	Specified class is either abstract or is an interface and must be concrete before use.	Verify the application class.
7006 - FileNotFoundException	The file either does not exist or is inaccessible (that is read only when trying to write).	<ul style="list-style-type: none">■ Verify the file's existence on the file system.■ Verify the file's permissions.
7007 - NoSuchMethodException	The method does not exist.	Verify the application class.
7008 - UnknownHostException	The host and the IP Address could not be resolved.	Contact your network administrator.
7010 - unable to get an instance of the class	See 7001.	See 7001.

Web Generation Flow Manager Error Messages 7100-7199

Exception	Reason	Action
7100 - A NULL procedure step name could not be located.	Unknown.	None. Fatal error.
7101 - Unable to locate procedure step class.	The procedure step class either does not exist or is inaccessible.	<ul style="list-style-type: none">■ Verify the existence of the procedure step class on the file system.■ Verify the permissions of the procedure step class.
7110 - A Process Action Diagram USE statement of a specified procedure step failed.	The procedure step class either does not exist or is inaccessible.	<ul style="list-style-type: none">■ Verify the existence of the procedure step class on the file system.■ Verify the permissions of the procedure step class.
7111 - View mapping failed.	The view types are either incompatible or do not exist.	<ul style="list-style-type: none">■ Verify the views in the model.■ Verify the views on the file system.

Web Generation Servlet Manager Error Messages 7200-7299

Exception	Action
7200 - A Fatal Error has occurred	None: This message is for error building. Localize as needed.
7201 - For URL:	None: This message is for error building. Localize as needed.

Chapter 9: Unsupported Features

This chapter is intended to assist users in converting their existing CA Gen Windows GUI clients to the new Web Generation 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
- Browser Differences
- Web Graphics
- Colors
- MAKE Support
- Asynchronous Support
- Help Support

This section contains the following topics:

[Unsupported Events](#) (see page 66)

[Unsupported Presentation-Related Functions](#) (see page 67)

[Unsupported OLE Functions](#) (see page 70)

[Unsupported File-Related Functions](#) (see page 70)

[Other Unsupported Features](#) (see page 71)

[Window/Dialog Properties](#) (see page 72)

[Browser Differences](#) (see page 73)

[Web Graphics](#) (see page 73)

[Colors](#) (see page 73)

[MAKE Support](#) (see page 74)

[Asynchronous Support](#) (see page 75)

[Help Support](#) (see page 75)

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	RightMouseDown RightMouseBtnUp WinMove WinResize
Single or multi-line entry field	RightMouseDown RightMouseBtnUp MouseMove Keypress
Check box	RightMouseDown RightMouseBtnUp MouseMove
Radio Button	RightMouseDown RightMouseBtnUp MouseMove
Non-Enterable Drop down	LeftMouseDown LeftMouseBtnUp RightMouseDown RightMouseBtnUp MouseMove
Non-Enterable List	RightMouseDown RightMouseBtnUp MouseMove MouseEnter MouseExit GainFocus LoseFocus
Picture	RightMouseDown RightMouseBtnUp MouseMove

Unsupported Presentation-Related Functions

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

- Beep
- BackgroundColor
- ClearDisplayProperties
- ForegroundColor
- GetCaption
- GetHandle
- GetHeight
- GetLeft
- GetPromptHeight
- GetPromptLeft
- GetPromptTop
- GetPromptWidth
- GetTop
- GetVisible
- GetWidthMessageBoxBeep
- Mark Command
- Refresh
- RestoreDisplayProperties
- SaveDisplayProperties
- SetCaption
- SetFont
- SetFontDynamically
- SetHeight
- SetLeft
- SetPromptHeight
- SetPromptLeft
- SetPromptTop
- SetPromptWidth

- SetTop
- SetVisible
- SetWidth
- Unmark Command

Tips

Tips for the unsupported presentation-related functions are:

- 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.
 - Dot notation cannot be used with Hypertext links the same as with HTMLText and HTMLControl, which were introduced in 6.5.
 - 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 Application 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 function MessageBox can only be used in Web Applications after the primary window or primary dialog has been displayed. This function must not be invoked from Procedure Step action logic, from Open or Activate event actions, or from TIREVENTs, which are invoked before the window is displayed.
- 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 desired effect.
- Additionally, some CA Gen Interface Object Methods and Properties are represented internally by Get or Set functions. Thus, they are not supported. The following CA Gen Interface Object Properties are not supported:
 - Caption
 - Enabled
 - Focus
 - FontSize
 - FontStyle
 - FontType

- Groupbox Caption
- Handle
- Height
- Left
- Literal Value
- Maximized
- Minimized
- Prompt
- Prompt FontStyle
- Prompt FontType
- Prompt Height
- Prompt Left
- PromptFontSize
- PromptTop
- PromptWidth
- Top
- Visible
- Width
- The following CA Gen Interface Object Methods are not supported:
 - Clear
 - Click
 - 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, since Object Linking and Embedding (OLE) technology is not supported by browsers:

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

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 also use OLE objects, which are not supported.

Even though file functions, such as OpenTextFile, are enabled in Web View environments, they are not thread safe. Due to the inherent multi-threaded nature of Web Applications, care must 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 potentials for collisions. Furthermore, references to these resources may not be preserved from one request to the next. You will 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 must create a unique file per user session.

Where not supported, action diagram statements are ignored at runtime.

Other Unsupported Features

The following functionalities are unsupported, or partially supported in Web Generation:

1. Due to the limitations imposed by HTML, the following controls are not supported:
 - OLE Area
 - Enterable DropDown List
 - Enterable List
 - ASP.NET controls
2. The following functionalities associated with specific controls are not supported:
 - DisableBy for Active X controls
 - Auto tabbing among Entry Fields
 - Extended Selection is only supported for list boxes on Internet Explorer
 - Accelerators for push buttons and Menu or Menu Items

3. The following table discusses the functioning of *Disabled By* logic in Mozilla Firefox and Internet Explorer:

Operation	Mozilla Firefox	Internet Explorer
Cut, Paste, Delete, and Undo from the Keyboard consistently (without using the Context menu)	<i>Disabled By</i> works as expected.	<i>Disabled By</i> works as expected.
Cut, Paste, and Undo from the Context menu consistently (without using the keyboard)	<i>Disabled By</i> does not work as expected.	<i>Disabled By</i> works as expected.
Delete by using the Context Menu	<i>Disabled By</i> does not work as expected.	<i>Disabled By</i> does not work as expected.
Delete key on the keyboard to delete the contents of a control and then use Undo.	<i>Disabled By</i> does not work as expected.	<i>Disabled By</i> does not work as expected.
Select the data to be deleted, use the Delete key on the keyboard to delete the contents of a control, and then use Undo.	<i>Disabled By</i> does not work as expected.	<i>Disabled By</i> works as expected.

4. Tab sequencing among window controls is governed by the controls' order in the generated HTML file. Since a list box is implemented by HTML <TABLE> and Link (Action:) tags, sequencing does not include list boxes.
5. Recursive calls within action blocks (such as, an action block calling itself) are not supported and may result in runtime abends.
6. The HideRadioButtonGroup method is not supported in Web Generation. Use the Visible property to hide the Radio Button Group controls.

Window/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.

All the HTML pages for a given window or dialog have a common appearance based on a predefined style sheet (Cascading Style Sheet). Custom Cascading Style Sheets (CSS) are not supported in Web View.

The title property of the window or dialog is used in the <TITLE> element of all HTML files generated for that window or dialog. The icon file associated with the window or dialog is not supported and the property is ignored.

Browser Differences

Due to the differences between browsers, if one browser only supports a given functionality, the implementation with the more restrictive functionality may be used to avoid illegal page content.

Browsers also differ in their implementation of style sheets and the order of event firing. Therefore, the same CA Gen web client windows may look and behave differently, depending on the browser and browser version.

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 Support

The MAKE statement's functionality is only partially supported for Mozilla Firefox and Internet Explorer. Note that MAKE support may not be effective when dealing with imported HTML pages that were created using third-party web authoring tools external to CA Gen.

MAKE in Internet Explorer

The following table illustrates CAGen's 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	No	No	Yes **	No	Yes **	Yes	Yes
Radio Button	No	No	Yes **	No	Yes **	No	Yes
Drop down List	Yes	Yes	Yes	Yes	Yes	Yes	Yes
List box (implemented with HTML tables and links)	Yes	Yes	Yes	Yes	Yes	No	Yes *

Note: ** Color changes are on the controls, not on their prompt.

MAKE in Mozilla Firefox

The following table illustrates CAGen's support for MAKE in Mozilla Firefox:

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	No	No	No	No	No	Yes	Yes
Radio Button	No	No	No	No	No	Yes	Yes
Drop down List	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Control	Color	Highlight Underline	Highlight Reverse	Intensity Dark	Error	Cursor	Protect
List box (implemented with HTML tables and links)	Yes	Yes	Yes	Yes	Yes	No	Yes*

Note: * Protected Fields in a List Box are placed within <P> and </P> tags. As a browser feature, when you click on such protected text, the browser will not highlight it. However, if you double-click, Ctl+click or Shift+click, Internet Explorer will highlight the text while Mozilla Firefox will highlight the border of the cell. Events associated with protected rows will not be fired.

Asynchronous Support

Web Generation does not support Asynchronous Processing.

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 10: JDBC Drivers

Web Generation is capable of supporting DBMS access from the action blocks running on the Application Server machine. This type of access is made using JDBC calls through a JDBC driver. It is possible to access databases not normally supported by CA Gen if a proper driver exists for them.

CA Gen works with any driver for JDBC 2.0 or later. It also works with type 1, type 2, type 3, and type 4 drivers. The steps required for installing drivers are similar, with some differences depending on the driver type. A type 4 driver is assumed in the following example.

To properly configure CA Gen to work with a JDBC driver, the following information is needed:

- Location (directory or jarfile) of the JDBC driver. (such as, d:\db2\java\db2java.zip for DB2)
- JDBC driver class name (such as, COM.ibm.db2.jdb.app.DB2Driver for DB2)
- DBMS URL for the driver (such as, jdbc:db2:myDB for DB2)
- userid
- password

DDL Installation

The table definitions may be installed in a variety of ways:

- Manually.
- With CA Gen, using the platform and DBMS specific generation options, for instance, generates for UNIX, Oracle and installs with the CA Gen installation program.
- With CA Gen, using the JVM platform. When installing this way, the Build Tool performs the installation using a Java JDBC installation program. When generating the DDL, the selection of DBMS specifies to the DDL generator which namespace from the Technical Design must be used in the DDL.

Note:

- A JDBC DBMS in the Technical Design for naming objects will be used in a DBMS that is not supported.
- If a DBMS is used to support traditional CAGen applications along with Web Generation or Java applications, it is important to duplicate the table and column names used in that DBMS.
- When installing DDL using the JDBC approach, the actual database must already exist within the DBMS. JDBC drivers are incapable of performing any type of CREATE DATABASE command.

Code Generation

To generate Java Web Generation applications that perform JDBC access, use the following settings in the generation environment:

- OS—JVM
- DBMS—JDBC or a specific DBMS (see the DDL Installation section for the reasons for selecting a specific DBMS)
- TPMONITOR—INTERNET

Note: Even though the target OS is JVM, the actual build can only be performed using the Windows Build Tool. After the build is complete, the Java class files may then be moved to any target operating system.

How to Configure the Target Environment

Two methods are available to configure JDBC access. The first relies on the DriverManager class to create connections, while the second uses DataSource objects to create connections.

DriverManager Method

This method is simple to set up. However, it does not allow for two-phase commits, Application Server connection pooling, or participating in Java Transaction Architecture (JTA) transactions. To use the DriverManager approach in Java applications, the following steps must be performed to set up the Application Server machine:

- The JDBC driver location (directory or jar file) must be loadable by the Application Server's Class Loader. This can be through an Application Server setting, copying the drivers to a specific location, or modifying the Classpath. See your Application Server documentation for details.

- The `jdbccfg.properties` file (installed in the CA Gen directory) must be loadable by the Application Server's Class Loader. This can be through an Application Server setting, copying the drivers to a specific location, or modifying the Classpath. See your Application Server documentation for details. The `jdbccfg.properties` file must be modified to provide additional configuration information to the runtime.

For each DBMS to be accessed from all applications running under the Application Server, some configuration properties should be added. Each property name is created using the database name, as the CA Gen Model understands it. This name may not be the name of the actual physical database, but is treated more as a logical database name to the application.

Any of the following four tokens in the `jdbccfg.properties` file may be specified for each logical database (where `xxxxx` represents the logical database name):

- (Required) `xxxxx_driverManager`—Classname of the JDBC driver to be used.
- (Required) `xxxxx_dataSource`—Datasource or URL name of the database to connect to.
- (Optional) `xxxxx_userId`—User ID to be used for connection.
- (Optional) `xxxxx_password`—Password to be used for connection.

The following are some examples using a database named `GENDB` and an Oracle driver:

- `GENDB_driverManager`—`oracle.jdbc.driver.OracleDriver`
- `GENDB_dataSource`—`jdbc:oracle:thin:@host:1521:ORCL`
- `GENDB_userId`—`scott`
- `GENDB_password`—`tiger`

DataSource Method

This method requires an Application Server-specific setup. It allows for two-phase commits, Application Server connection pooling, and participating in Java Transaction Architecture (JTA) () transactions.

To support DataSources in Java applications, the JDBC driver location (directory or jar file) must be loadable by the Application Server's Class Loader. This can be through an Application Server setting, copying the drivers to a specific location, or modifying the Classpath. See your Application Server documentation for details.

To use the JDBC DataSource method to create connections, the Application Server uses the JNDI service provider. Therefore, the JDBC connection information must be specified to the Application Server as a resource.

Note: For specific information about how to set up resource adapters, see the *Distributed Processing – Enterprise JavaBeans User Guide*

Chapter 11: Cross-Context Flows

There are situations where deploying an application in a single ear file limits the flexibility of developing, deploying and maintaining that application. Cross-context flows can improve that flexibility considerably; however, there is a trade-off in application performance.

Cross-Context Flow

A Cross-Context Flow occurs when the procedure steps participating in a flow are deployed in separate web applications. A Procedure Step USE, where the procedure steps are in different web applications, is also considered a Cross-Context Flow. We will refer to Links, Transfers, and USE operations between procedure steps deployed in separate web applications as Cross-Context Flows.

There is no difference in the modeling of the flow or the procedure step USE in a cross-context application, from the modeling done for a single-context application. The difference is largely a factor of application Assembling and Deployment.

The web applications must be generated and assembled from the same Gen release and must be deployed under the same versions of the JVM and Web Application Server.

This implementation of flows for clients has no effect on client-server flows. Flows to or between servers are also unchanged.

Using Cross-Context Flows

The primary usage of Cross-Context Flows will be for Component Based Development, where transactional components are deployed independently.

Note: For information about Component Based Development, see the *Component Standard 3.1.1* white paper.

Other usage of Cross-Context Flows could occur:

- To override one part of a deployed ear file to facilitate change or testing.
- To transition from one application to another.
- To isolate separate parts of the application for different models and different support groups.

Performance Impact

A Cross-Context Flow is handled by sending required data between applications participating in Cross-Context Flows using a `URLConnection`. This data is necessary for transmitting state information and application data such as Views. This data is transmitted using serialization across the connection. The process of serializing and de-serializing the data, along with the communication across the connection, causes performance degradation.

In addition, for Links and Transfers that cause a display from the flowed-to context, the data must be stored to a Java Message Service (JMS). This makes the data available when the browser is redirected to the flowed-to context. This additional storage and retrieval of data will also impact performance.

The larger the view sizes being sent between procedures, the more significant this impact will be. While this may not be a problem when testing a new element destined to be ultimately deployed in the same web application, it should be considered in a system architecture that plans production use of Cross-Context Flows.

Java Message Service (JMS) Session Storage

When a browser is sending requests to an application, a session is created between the browser and the application. When doing Cross-Context Flows in which the second context requires a display, the browser must be redirected to the application running under the second context. This redirection causes a new session to be created between the browser and the second application.

Since there are two separate sessions created by the application server(s), the second application does not have direct access to the session information of the first application.

To complete the transfer, the session information from the first application must be transferred to the second application. The session information is transferred using a `URLConnection` and stored in a Java Message Service (JMS) while the browser is redirected to the second application. When the browser and the second application initiate a session, the session information is reloaded from the JMS.

For this technique to work, the JMS must be configured on each Application Server that will participate in Cross-Context Flows. In addition, the JMS connection factory name and queue name must be configured during assembly in the Build Tool.

Note: The Assemble process only supports JMS configuration for WebLogic and WebSphere Application Servers. If Cross-Context Flows are to be used with other Application Servers, then Application Server specific customizations will have to be made to the EAR file after the Assemble process has completed. These customizations will be in addition to the usual JMS configuration steps that are required on the Application Server itself.

Map Load Modules

URLs are used to reference all load modules in the CA Gen Java runtime. Most of the time, for backward compatibility, a URL value is null which indicates the Load Module must be found in the current application context. When Cross-Context Flows are desired, the URL must be set to the protocol, server, and context that support the actual Load Module to be executed.

The relationship table of Load Module name to URL is created by the CA Gen Build Tool's Assembly and generated into the web.xml file of each WAR included in the application EAR. Within the file web.xml, each Load Module name is prefixed by *url/* to become the name of an environment entry whose value is the URL that supports the Load Module.

Each time a procedure step is loaded, the CA Java Runtime looks in the current load module for the procedure step. If the procedure step is not found in the current load module, then the URL for the appropriate load module is determined from the web.xml table. The Remote Load Module Exit, WindowManagerCfgExit, is then called to allow for programmatic overriding of the URL. If it is determined that the requested Load Module is in a different context, then the URL is used to access the remote context.

How to Configure a Cross-Context Flow

To utilize Cross-Context Flows, configuration of Load Module mappings must be done. In addition, a Java Message Service (JMS) must be configured in the Application Server. The Build Tool allows for configuration of the Load Module mapping along with the JMS connection factory and queue names.

An application may have all of its load modules assembled together in a single context, or the load modules may be separated into independently deployed contexts. You should keep in mind; however, that communicating across separately deployed context will have a negative impact on performance.

It is recommended to name your load modules in a way that enables you to identify them quickly when configuring for cross-context. For example, including an identifier for the model in the load module name will help you identify the local and remote load modules at assembly time. For example, if you have two CBD component models, you might name your load modules with CBD1LM1, CBD1LM2, CBD2LM1, and CBD2LM2. This makes it very clear at assembly time which load modules come from CBD1 and CBD2.

There are two main methods of configuring the Load Module mapping between procedure steps that reside in different contexts:

- **Build Tool Configuration**—lets the user define the URLs of Load Modules that belong to procedure steps in different contexts. A list of all Load Modules referenced by the currently selected Load Modules will be displayed for configuration. The Load Module mapping must be set to the full URL of the Application Server Context

For example:

`http://www3.ca.com/press`

- **WindowManagerCfgExit.java**—is a source file whose default behavior returns the URL as it is passed. This exit lets the application designer define a mapping algorithm between load modules and URLs. Java code can be used to override the URL mappings that were defined during the Cross-Context configuration when assembling the application. The compiled version of `WindowManagerCfgExit.java`, `WindowManagerCfgExit.class`, will be included along with all other user exits into the runtime jar file as built by the Assemble process.

Design Considerations

When configuring components in different contexts to communicate with one another, there are several areas you need to consider:

- The Dialect used in the component you first invoke will be the one used in every subsequent component. You cannot switch dialects when going from one context to another.
- Browser Controlled History Handling is not supported for Cross-Context Flows and the Build Tool will turn this feature off if Cross-Context Flows are enabled.
- The domain name, *localhost*, cannot be used for Cross-Context Flows. This includes both the Load Module mapping that is done in the Build Tool and `WindowManagerCfgUserExit`, as well as the actual initiating URL address entered in the browser.

- A procedure step that is invoked through a Procedure Step Use may not flow to another procedure step that causes a display.
- Database transactions cannot be extended across multiple contexts. Each flow across context boundaries will cause a new transaction to be created for the duration of that procedure step. In the case of a USE, the current transaction is suspended and a new transaction is created for the duration of the called procedure step. Upon return, the original transaction is continued. Care must be taken to avoid deadlock conditions when processing USED procedure steps.

Following are examples you can consider to reduce the performance degradation you will experience when using Cross Context Flows:

- Applications utilizing Cross Context Flows will be slower than those that stay within the same context. You may consider using Cross Context Flows during your Development cycle and for Production, package your application in a single context.
- Cross Context Flows between separate applications deployed on the same local application server system are likely to perform better than those that are deployed on separate systems. This is because the data will not be transferred across the network.
- The larger the sizes of the views that are passed across contexts, the bigger the negative impact on performance will be. Consider reducing the sizes of views that will be transmitted across contexts.
- Chaining Link flows will cause the amount of information passed between each context to grow. For example, doing Links from PStepA to PStepB to PStepC will cause the session information for both the PStepA and PStepB to be passed along to the context in which PStepC is located. Avoid Chaining Links across contexts.
- If possible, use Transfers instead of Links when flowing across contexts.
- Configure the Application Server's JMS to cache the session messages in memory.

Index

A

activeX controls, embedding • 36

C

CA Gen • 77

 creating databases • 77

 DDL installation • 77

Cascading Style Sheets • 29

common edit modifications • 20

Common video properties • 23

configuring JDBC access • 78, 79

 DataSource approach • 79

 DriverManager approach • 78

configuring web generation communications • 54

 caching • 54

 format for trancodes • 54

 tracing using CMIDEBUG • 54

consistency check messages • 13, 61

 client runtime • 13

 technical requirements • 13

CSS Class attributes • 21

Custom video properties • 23

D

DBMS access, using JDBC calls • 77

DDL installation, CA Gen • 77

E

error messages, Java web generation • 61

H

HTML, editing • 41, 49

 administration • 41

 regeneration of web generation applications • 49

HTML, using • 41, 42, 43

 cookie-free approach • 43

 handling ScrollTop and ScrollBottom events • 42

 using HTML pages without frames • 41

HttpServletRequest • 57, 58

I

installation, DDL • 77

J

Java web generation error messages • 61

JDBC access, configuring the environment • 78

P

pre-installation planning • 13

preparing the web generation application for web access • 13

R

release • 13

remote • 13

 technical requirements • 13

S

setting up the web generation

 environment, technical requirements • 13

T

target platform • 13

technical requirements • 13

third party software requirements • 13

V

Video Properties • 22

W

web generation, JDBC access • 78