

CTC ASP.NET WebForms Generator

Version 2.0.3



Table of Contents

1	Introduction	4
1.1	What is CTC ASP.NET WebForms Generator?	4
1.2	The Concept.....	4
1.3	Standard Controls	5
1.3.1	CTC Multi Level Context Menu	6
1.3.2	CTC ComboBox Control.....	7
2	Generator Initial Setup.....	7
2.1	EAE Setup.....	7
2.2	AB Suite Setup.....	9
2.3	Installing Bundle Infrastructure Files	10
3	Configuring the Generator	11
3.1	Generator Options.....	11
3.2	Generating CopyFrom As Table	17
3.3	Runtime Configuration.....	19
3.4	Control Specifications	19
4	The Generated User Interface Application	19
4.1	The Visual Studio Solution	19
4.2	The Generated Ispec Forms	20
5	CTC ASP.NET View Controller	22
5.1	CTC ASP.NET View Controller API.....	22
5.1.1	Events	22
5.1.2	Methods.....	23
6	Custom Controls.....	23
6.1	System Provided Custom Controls.....	23
6.2	Creating Own Custom Controls.....	26
6.2.1	Custom Controls – The Generate Side.....	27
6.2.2	Custom Controls - The Runtime Side.....	29

1 Introduction

1.1 What is CTC ASP.NET WebForms Generator?

The CTC ASP.NET WebForms Generator is a tool for creating ASP.NET WebForms user interface for EAE 3.3 and AB Suite systems. The generator provides support for ASP.NET AJAX as well as the standard ASP.NET.

The Generator is an add-in to the Unisys Component Enabler Generate Environment. The generated user interface application utilises the Unisys Component Enabler interface to communicate with the EAE and AB Suite host systems.

This document should be read in conjunction with the **CTC ASP.NET WebForms Configurator** document and the **CTC Configurator Framework** document.

1.2 The Concept

Typically, a generator creates a fixed, predetermined user interface application, where users have no or limited influence on what is generated, and customizations have to be applied by modifying the generated user interface application or by writing a custom generator.

The concept of CTC Generators is to include as many requirements as possible in the generate stage rather than applying modifications to the User Interface after it has been generated.

This requires the generator to be very flexible and to provide the means for customizing the result of the generator prior to entering the generate phase.

CTC Generators provide the ability to influence what is generated by configuring features, setting options, customizing Standard Controls, adding Custom Controls and substituting controls. The generated user interface is still based on the forms and controls being painted in the EAE or AB Suite development environment, however, the CTC ASP.NET WebForms Configurator allows the developers to specify how each form and control is to be generated at the application, bundle, language, ispec and field level.

Being able to configure and specify the required customization before the generate phase provides a repeatable and automated solution that in most cases will not need further modifications to the generated source.

In order to provide the necessary flexibility, the Generator includes a library of Standard Controls, one control for each control that can be painted in the EAE and AB Suite development environment. A Standard Control is a self-contained type that understands exactly how to interpret the information from the painted control and how to create itself as the equivalent ASP.NET Web Form Control. This allows the appearance of each control to be easily configured and, when necessary, Custom Controls can easily be created by extending the Standard Controls through inheritance. Custom Controls that map to other ASP.NET controls or third party controls can be created.

When the user interface is being generated, the generator receives the information from the EAE and AB Suite development environment. For each ispec, it creates a form control, adding each of the controls to the collection of controls on the form. The form and the controls in the collection generate themselves as the corresponding ASP.NET controls. During the process, configuration information such as specific control properties or the replacement of Standard Controls with Custom Controls is applied to the form and the controls.

1.3 Standard Controls

Standard Controls provide the default look and feel of the controls as they are painted and specified in the EAE and AB Suite Development environments. Standard Controls are built into the ASP.NET WebForms Generator. They can be used as they are without further customization. However, they can also form the basis for customization.

The following is the list of Standard Controls available with the ASP.NET WebForms Generator:

Control	Description
ButtonGroup	Container for single button controls such as Check Box, Push Button and Radio Button. AB Suite only.
CheckBox	Single button Check Box. AB Suite only.
CheckBoxList	List of Check Boxes, horizontal/vertical. EAE 3.3 only.
ComboBox	Drop down list box.
Form	Container for controls painted on a form.
Form Page	Outermost container for the form.
Image	Image.
Label	Label.
Line	Box and horizontal and vertical line.
ListBox	List Box.
ListDataSource	DataSource control for ListBox and ComboBox controls.
ListXmlDataSourceControl	DataSource control for static ListBox and ComboBox controls.
Panel	Group container for other controls. AB Suite only.
PushButton	Single Push Button. AB Suite only.
PushButtonList	List of Push Buttons, horizontal/vertical. EAE 3.3 only.
RadioButton	Single Radio Button. AB Suite only.
RadioButtonList	List of Radio Buttons, horizontal/vertical. EAE 3.3 only.
Table	Table container for CopyFrom regions.
TableHeaderRow	Table Header Row container for CopyFrom regions.
TableHeaderCell	Table Header Cell container for CopyFrom regions.
TableRow	Table Row container for CopyFrom regions.
TableCell	Table Cell container for CopyFrom regions.
Teach	Container for controls painted on a teach form.
Teach Page	Outer container for the teach form.
TeachText	Text for the teach form.
TextBox	Text input box.

1.3.1 CTC Multi Level Context Menu

Included with the CTC ASP.NET WebForms Generator is a context menu control, which provides a number of properties unique to CTC.

The Context Menu control allows creating a true context menu that reflects the actual context in which the browser user requests the menu. Menu items can be added and removed dynamically in the code behind.

The Context Menu is a control that can be styled at design time to suit local requirements using Visual Studio. The control is defined on the Default.aspx page. Therefore, to style the Context menu, open the Default.aspx page in Visual Studio, select the control and set the properties.

The BoundControlList property allows the specification of a list of ViewController controls for which the context menu is displayed. This property is used to bind one or more controls to a context menu control. It is possible to add two or more context menus to a page and in this case the BoundControlList property specifies which controls on the page each of the menus are bound to.

The whole context menu as well as individual menu items can be enabled and disabled.

Other properties such as RolloverBackColor, RolloverForeColor, DisabledBackColor and DisabledForeColor are available.

Multi level sub menus can be added to the Context Menu at design time, using the ContextMenuItems property, or dynamically at runtime. For example, this allows the adding of a list of ispecs as a sub menu to the Context Menu providing an 'ispec quick pick' function for the end users.

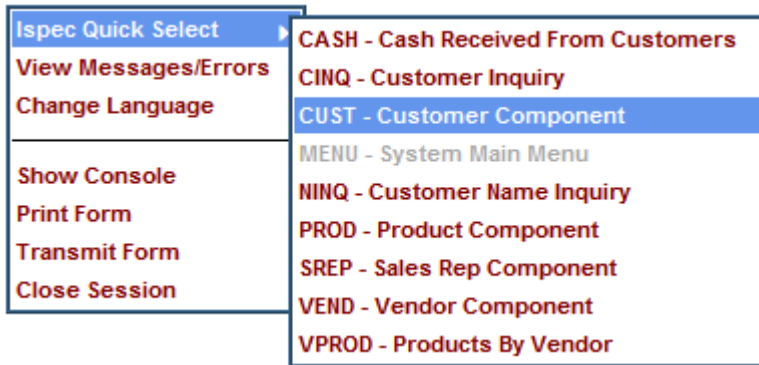
In order to provide an efficient context menu, the menu is only created and sent to the browser when requested by the browser user. This keeps the amount of html code sent to the browser to a minimum.

At runtime, the Context Menu control raises a number of events which the hosting web page can choose to listen to and handle:

- **CreateContextMenu:** This event is raised as a result of the browser user right-clicking on the page requesting a context menu. This allows the hosting page (see Default.aspx.cs) to create the menu, and add, modify and delete items on the context menu and respond to the user with a menu that is relevant to the specific context.
- **RestoreContextMenu:** This event is raised when the EnableViewState property on the Context Menu control is set to false. In this case, the Context Menu Control doesn't maintain its state automatically and it is therefore necessary for the hosting page to restore the state of the context menu before processing a postback as a result of the user selecting an item on the menu.
- **SelectContextMenu:** This event is raised as a result of the browser user selecting an item in the context menu. The hosting page must process the request.

See the Default.aspx.cs for a comprehensive example of how to create a context menu that responds to the browser user's request, dynamically builds a list of ispecs currently generated for the bundle and adds it as a sub-menu to the standard context menu. The Default.aspx.cs contains a property named 'EnableIspecListSubMenu' which allows the user to easily enable/disable the creation of an 'ispec quick pick' sub-menu. The property is enabled by default. This example can be further enhanced to suit local requirement.

The following is an example of a multi level Context Menu:



The id of the control which the browser user right clicks on when requesting the context menu is passed into the hosting web page via the events as described above. This provides the opportunity for the hosting web page to add items to the context menu that are specific to the control.

The Default.aspx.cs also includes an example of how to add an item to the context menu that allows the user to copy list data to the clipboard in excel format. When the browser user right clicks on a ListBox, GridView, DataList, CopyFrom Table or control within a CopyFrom region, the item "Copy to Clipboard, Excel Format" is added to the menu. Subsequently, when the user selects the "Copy to clipboard" item, the data from the list is copied to the clipboard in excel format ready to be pasted into an excel spreadsheet. The Default.aspx.cs contains a property named 'EnableCopyToClipboardMenu' which allows the user to easily enable/disable the creation of this menu item. The property is enabled by default. This example can be further enhanced to suit local requirement.

1.3.2 CTC ComboBox Control

Included with the CTC ASP.NET WebForms Generator is a ComboBox control, which provides a number of properties unique to CTC.

AutoComplete property, when set, automatically completes the text field with the value from the list that matches the entered text as the user enters text.

DropDownStyle property allows the specification of the style of the dropdown - Simple, Dropdown or DropDownList.

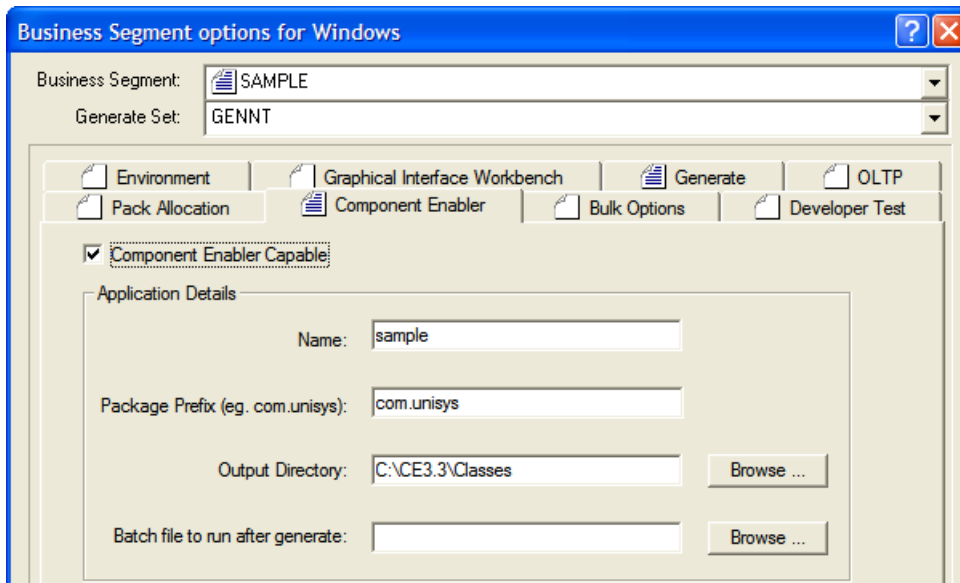
TextBoxBorderStyle specifies the border style of the TextBox part of the ComboBox.

2 Generator Initial Setup

The CTC ASP.NET WebForms generator is an add-in generator to the Component Enabler Generate Environment and as such, the setting up of the generator follows the standard instructions for any CE Compliant generator.

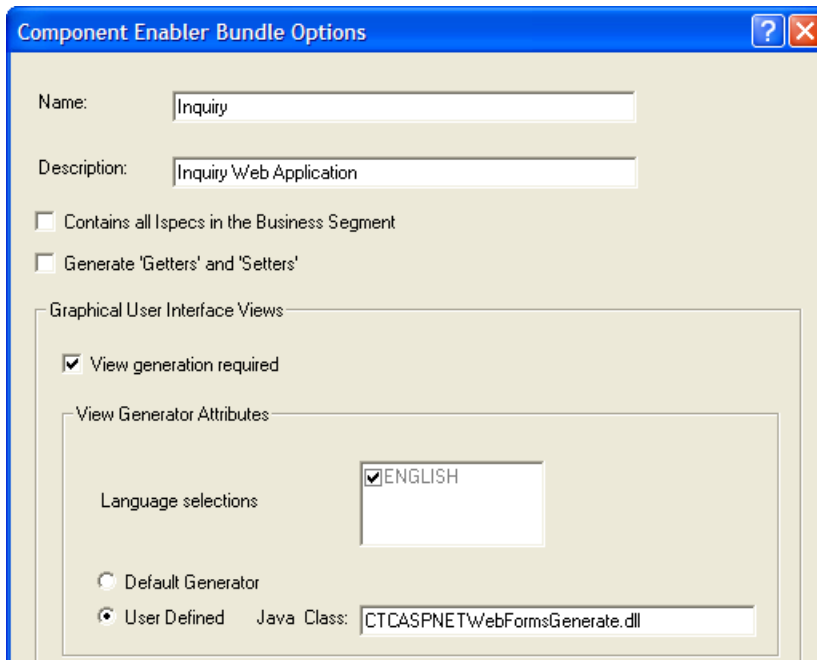
2.1 EAE Setup

Within EAD (Enterprise Application Developer), Application Details must be specified for Component Enabler in the Business Segment Options dialog as shown in the following dialog.



Together with the Bundle Name specified in the following dialog, Application Name, Package Prefix and Output Directory define the path to output location of the generated user interface application. The path is [OutputDirectory]\[PackagePrefix][ApplicationName][BundleName], i.e. the output location for this example would be C:\CE3.3\Classes\com\unisys\sample\inquiry.

For each bundle to generate, the bundle details must be specified in the Component Enabler Bundle Options dialog as shown below.



The name of the generator to use must be entered in the Java Class field. The name of the CTC ASP.NET WebForms Generator must be specified exactly as shown. Generators from CTC are implemented using .NET and the C# language, hence the .dll extension on the name. The reference as specified above is a relative reference to the [ceroot]\bin directory, which is where the generator is located when installed.

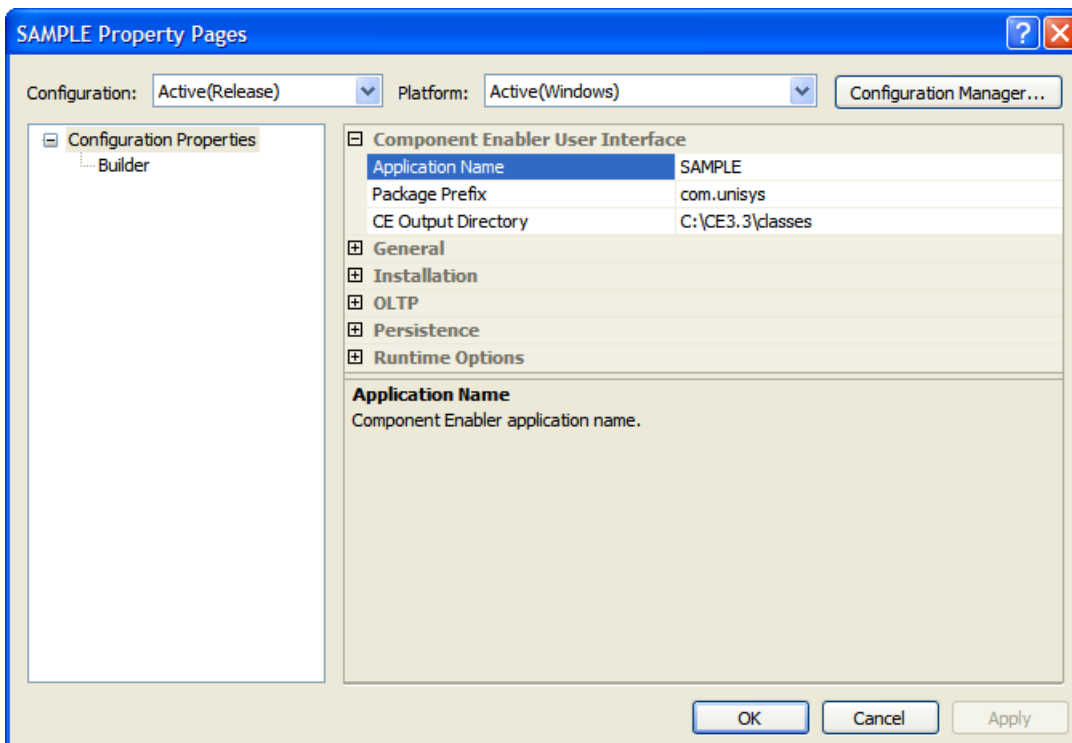
In addition to installing the CTC Silverlight Client Generator, users of EAE 3.3 IC3260 or earlier must also install the CTC Generate Gateway. The CTC Generate Gateway provides the necessary interface allowing the EAE Generate Environment to invoke generators implemented in .NET. Users of EAE 3.3 IC3270 or later must be using CE 2.0 with the parameter UseDotNET in linc.ini set equal to Y. For further information see the ReadMe file of EAE 3.3 IC 3270.

Ispc Models must be generated and compiled for C# and .NET. Prior to EAE 3.3 IC 3210, ispec models were generated for C# but had to be manually compiled. From IC 3210 and later, ispec models can be automatically compiled by adding 'GenerateCSharpIspcModels=Y' to the Component Enabler section of the LINC.INI file. In addition it is recommended to also set GenerateJavaIspcModels=N in linc.ini. For further information see the ReadMe file of EAE 3.3 IC 3210.

For a full description of all fields and how to set up and add ispecs to a bundle, refer to the Component Enabler User Guide.

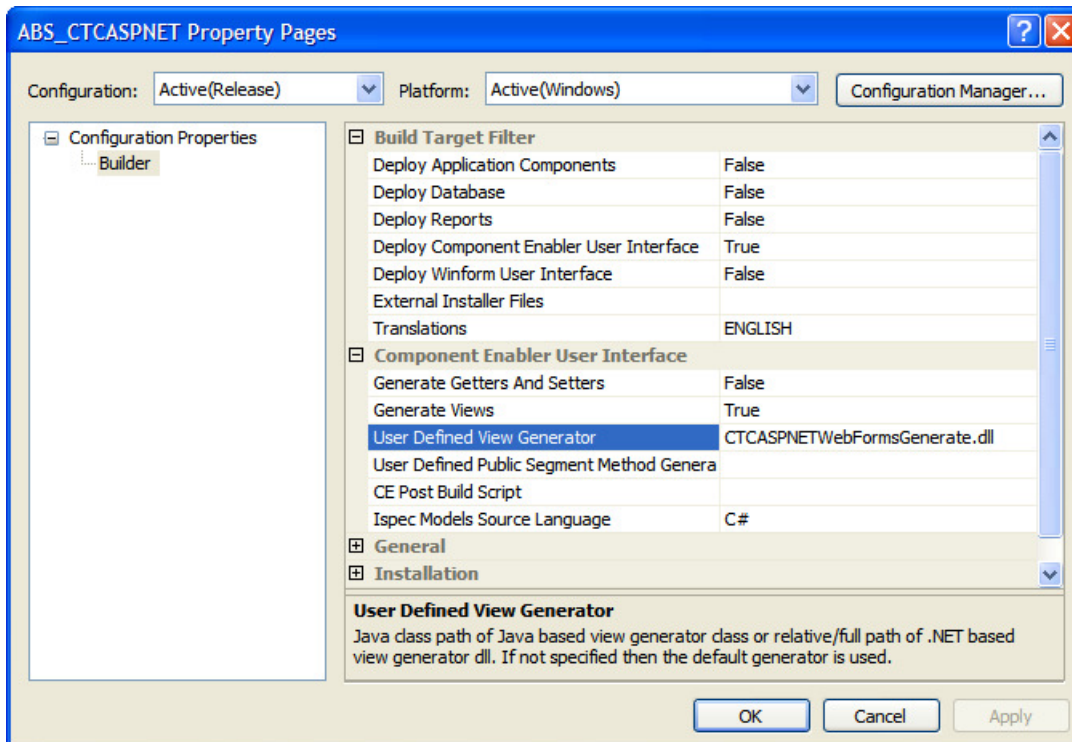
2.2 AB Suite Setup

Within AB Suite Developer, Application Details must be specified for Component Enabler in the Property Page dialog of the Business Segment Class as shown in the following dialog.



Together with the name of the folder defining the bundle as shown below, Application Name, Package Prefix and Output Directory define the path to output location of the generated user interface application. The path is [OutputDirectory][PackagePrefix][ApplicationName][BundleName], i.e. the output location for this example would be C:\CE3.3\Classes\com\unisys\sample\abs_ctcaspnet.

A folder defining the ispec classes to include in a bundle is added to the business segment. In the Property Page dialog for the folder, details for the bundle are specified as shown in the following dialog.



Deploy Component Enabler User Interface must be set to True.

The name of the CTC ASP.NET WebForms Generator must be specified exactly as shown in User Defined View generator. Generators from CTC are implemented using .NET and the C# language, hence the .dll extension on the name. The reference as specified above is a relative reference to the [ceroot]\bin directory, which is where the generator is located when installed.

Users of AB Suite are not required to install the CTC Generate Gateway as the AB Suite Generate Environment already provides the necessary interface for invoking generators implemented in .NET.

Ispc Model Source Language must be set to C#.

For a full description of all fields and how to set up and add ispecs to a folder/bundle, refer to the Unisys AB Suite User Guide.

When building the folder/bundle from AB Suite Developer it is recommended to always choose the 'Rebuild' option in order to ensure the configuration setting of the CTC Generator takes effect on all ispecs in the folder/bundle.

2.3 Installing Bundle Infrastructure Files

To keep the amount of code to be generated to a minimum and to provide a high level of flexibility at runtime, the generated user interface application requires a number of fixed non-generated files to be present in the output directory for compiling and running the application. These files are collectively referred to as Infrastructure Files.

No manual steps are required for copying the files. When running the generator for the first time on a bundle, the infrastructure files are automatically copied into the output directory of the bundle. The files to be copied are controlled by the 'CTCASPNETWebFormInfrastructureFiles.xml' file located in the [ceroot]\bin directory.

When any of the infrastructure files have been updated or new files have been added, the files are re-installed to the bundle by setting the Generator option `ReInstallBundle` equal `true`.

When generating a bundle for the first time after upgrading to a new version of the generator, infrastructure files that have changed will automatically be re-installed to the bundle.

When initializing a new bundle the generator automatically creates a virtual directory for the generated web application on the local host. By default the name of the virtual directory is `[ApplicationName]_[BundleName]` and it is mapped to the views directory of the bundle. The name of the virtual directory can be changed using the `VirtualDirectoryNew` configuration parameters. The generated application is run from the browser using [http://localhost/\[ApplicationName\]_\[BundleName\]/](http://localhost/[ApplicationName]_[BundleName]/) as the url.

When the virtual directory is created or managed manually, the name of the virtual directory must also be specified in the Visual Studio project for the generated forms.

3 Configuring the Generator

The generator is configured using the CTC Configurator. The CTC Configurator provides a flexible way of configuring the generator and the controls at any level in a hierarchy consisting of application, bundle, language, ispec and field. The higher in the hierarchy an option or control is configured, the more general it is specified. All lower levels in the hierarchy inherit the specification from the levels above. The lower in the hierarchy an option or control is configured, the more specific it is.

Because the generator may update the configuration file, it is recommended to close the CTC Configurator while running the generator.

For further details on how to use the CTC Configurator, refer to the **CTC Configurator Framework** and **CTC ASP.NET WebForms Configurator** documentation.

3.1 Generator Options

Alternate View Create

This option allows the creation of an alternate view for all or selected ispecs. When set, a copy of the current version of the generated Ispec View is created and added to the project.

Creating an alternate view using the view/form of the current generated ispec provides an easy starting point for designing forms using external tools such as Visual Studio WebForm's painter.

The name of the alternate view is added as an attribute to the list of ispecs being generated. This allows the main application to automatically switch to the alternate view when the end user navigates to an ispec for which an alternate view is specified.

Alternate View Remove

This removes the reference to the alternate view from the generated project. However, in order to preserve the alternate view, the files containing alternate view are not deleted. When later creating an alternate view again for the same ispec, the same files will be added to the generated project.

ASP.NET AJAX Control Toolkit Location

Microsoft provides a toolkit which includes a number of controls that extend selected standard controls with specific AJAX capabilities. This toolkit is available from the Microsoft Web Site as a separate download (see the CTC ASP.NET WebForms Generator ReadMe document for further details). Users who wish to take advantage of these controls and include additional AJAX capabilities in their user interface must download and install the toolkit.

This option specifies the location of where the toolkit has been installed.

ASP.NET Version

This option specifies whether to use 'ASP.NET 2.0 Standard', 'ASP.NET 2.0 AJAX', 'ASP.NET 3.5 Standard', 'ASP.NET 3.5 AJAX', 'ASP.NET 4.5 Standard' or 'ASP.NET 4.0 AJAX' when generating a bundle. Changing a bundle from ASP.NET Standard to utilizing AJAX and visa/versa is as easy as setting this option. Users who wish to take advantage of AJAX must download the Microsoft ASP.NET Extension which is available as a separate download from the Microsoft Web Site (see the CTC ASP.NET WebForms Generator ReadMe document for further details).

The generator will automatically copy infrastructure files appropriate to the chosen value to the bundle views directory. This includes files such as Default.aspx, Web.config and the generated project file.

Build Generated Solution

As part of the generate process, the generator can automatically compile the generated application and build the necessary dll's. This is achieved using MSBuild, which is the build platform from Microsoft used by Visual Studio. Details of the build are written to the generate log file.

CopyFrom As Table

When this option is set, the CopyFrom region of an ispec is generated as a Table with a row for each CopyFrom copy and each control is placed within a cell of the row.

The table is generated using the standard ASP.NET controls for Table, TableHeaderRow, TableHeaderCell, TableRow and TableCell. This provides the opportunity to configure and style the table and all its elements taking advantage of all the properties available with ASP.NET.

See the section 'Generating CopyFrom As Table' for further information.

CopyFrom Auto Column Headers

When set, the generator will look for Label controls painted on the form directly above the CopyFrom region and automatically place them as column headers within the Table.

See the section 'Generating CopyFrom As Table' for further information.

CopyFrom MultiLine As Single

When set, the generator will generate a CopyFrom spanning multiple lines as single lines. This option may be useful for MultiLine CopyFrom areas that are not suitable as table layout.

Custom Code Module Create

This option allows the creation of a module for all or selected ispecs in which specific custom code can be added. When set for an ispec, the generator adds a code-behind module to the generated form for the ispec and registers this with the generated project. Once created, the module is not affected in any way by the generator unless specifically removed using the 'Custom Code Module Remove' option.

The custom code module allows users to add code to handle specific requirements during the processing of the form. Setting properties on controls depending on values of fields returned from the host system and validating user input before the data is sent to the host system are examples of custom code that can be added to a form.

Within the Custom Code Module, users decide to implement one or more of four different methods which are invoked during the processing of a form.

- **BeforeUpdateForm():** This method is invoked just before controls on the form are updated with values of the fields returned from the host system as a result of a transaction to the host system. This allows the custom code to manipulate the controls before the form is presented to the user in the browser.
- **AfterUpdateForm():** This method is invoked right after controls on the form are updated with values of the fields returned from the host system as a result of a transaction to the host system. This allows the custom code to manipulate the controls before the form is presented to the user in the browser.
- **BeforeUpdateIspec():** This method is invoked just before fields on the ispec are updated with values from the controls returned from the browser as a result of the user submitting the form. This allows the custom code module to manipulate the ispec fields before the ispec is transmitted to the host system.
- **AfterUpdateIspec():** This method is invoked right after fields on the ispec are updated with values from the controls returned from the browser as a result of the user submitting the form. This allows the custom code module to manipulate the ispec fields before the ispec is transmitted to the host system.

Custom Code Module Remove

This removes the reference to the code-behind custom module from the generated project. However, in order to preserve the custom code, the file containing custom code is not deleted. When later creating a custom code module again for the same ispec, the same file will be added to the generated project.

Default Image Type

This defines the file extension to apply to image names when the image name returned from the host system doesn't contain an extension.

Default Date Format

This defines the default date format to apply to Date Controls.

Possible formats are:

- UK (ddMMyy, ddMMyyyy)
- US (MMddy, MMdyyyy)
- International (yyMMdd, yyyyMMdd)

Display Errors, Display Warnings

The user can be alerted to errors and warnings that occur during the generate process. By default, errors and warnings are displayed in a message box waiting for confirmation. Regardless of the setting of these options, errors and warnings are always written to the generate log file.

Enable View State

By default, all ASP.NET controls maintain their state by save state information in a hidden field that is passed in the page during round-trips between the web server and browser at runtime. Disabling the View State by setting 'EnableViewState=false' can save a considerable amount of data being transferred to/from the browser especially for controls like List boxes and Combo boxes.

Typically, EAE and AB Suite systems do not benefit from saving the view state, as a new form is always returned from the EAE and AB Suite host system for every transaction. Turning view state off by configuring this option will reduce the amount of data sent to and from the browser.

Exclude Teach Screen

When set, this exclude teach screens from the generate process.

Generate System Info

This specifies for the generator to write information about ispecs and fields contained in the bundle being generated to the CTCSystemInfo.xml file located in the [ceroot]/bin directory. This information is used by the CTC Configurator to populate the dropdowns on fields, ispecs, bundles and applications with valid selections making it easy to configure these items.

Identify GroupBox

This identifies Rectangles with Labels overlapping the top line of the rectangle as a GroupBox. This allows using the IsGroupBox expression in a MatchOnField expression for identifying rectangles and labels painted as a GroupBox.

IIS Reset

When set, the generator resets (Stop and Start) IIS before generating Ispec Models files. This avoids errors occurring during the generation of the Ispec Model files caused by IIS locking Ispec Models files that has previously been accessed.

Infrastructure Files Version Check

When re-initializing a bundle, this option determines whether to perform version check when copying infrastructure files to the bundle output directory. When set, only new and updated files are copied.

Keep Session Alive

For some web applications where data entry is a critical part of the application, it can be very disruptive to the end-user when the session with the web server times out while filling in the form and before the data is submitted. This option, which can be set for all or selected forms, prevents the session from timing out. When set, a background request is sent from the browser 30 seconds before the session is about to time out.

Label Id Detection, LabelIdStartSeparator, LabelIdEndSeparator

Labels painted on the form in EAE and AB Suite are not associated with data items and therefore they are not uniquely identified. This provides the means to identify labels so that, when required, it is possible to configure individual labels. A Label Id is specified within the text of the label enclosed in start and end separators.

ListBox Submit On Double Click

This specifies whether to submit the form to the server when the end user double clicks on an item in a list box.

ListBox Submit On Enter Key

This specifies whether to submit the form to the server when the end user hits the enter key on an item in a list box.

Log Level

This defines the level of detail written to the generate log file. The log information is written to C:\Temp\Generate.log.

Position Left Adjustment

This specifies a number of pixels to adjust (+/-) the left position of controls. When specified, the left position of controls this option applies to, will be modified with the value, which results in the controls being moved left or right in the horizontal direction.

Position Top Adjustment

This specifies a number of pixels to adjust (+/-) the top position of controls. When specified, the top position of controls this option applies to, will be modified with the value, which results in the controls being moved up or down in the vertical direction.

Re-Install Bundle

When set this option causes the generator to re-install the infrastructure files to the bundle output directory. This is required when new files have been added, when existing files have been updated or to repair damaged files. The re-install is performed when next starting the generate process of the bundle. When done, this option is automatically reset by the generator.

Remove Button Group Panel

When set this option removes the panel which is added by AB Suite as a group container for Button Groups when importing a model from EAE.

TextBox Auto Tab

Automatically tab to the next control in the tab order sequence when the maximum number of characters defined for the textbox has been entered.

TextBox Label When Read-Only

When the field painted as a textbox is defined as a read-only inquiry field, this option will create a Label control instead of the textbox.

TextBox Multi Line Max Length Check

Ensures the number of characters entered into a multi line textbox doesn't exceed the maximum number of characters defined for the field.

TextBox Validate Numeric

Ensures the value entered into a textbox is numeric for fields defined as numeric. The validation is performed according to numeric type and decimal character defined for the data field associated with the textbox such as LongSigned (CR/DR), Signed (+/-) or Unsigned.

Text Transparent

This option specifies whether to generate text controls such as Label and Panel with transparent background. When the background color of the form on which controls are placed has a background color that is different to the color of the controls, it may be desirable to generate labels and panels with a transparent background so they don't stand out as highlighted control.

Tool Tip

This option determines whether to apply the Help Text defined for the fields in EAE or AB Suite and display it as a tool tip when the mouse in the browser hovers over the control.

Virtual Directory Auto Create

This option configures the generator to automatically create a virtual directory for the generated ASP.NET application. On machines without IIS, this option should set to false.

Virtual Directory Name New

When initializing a new bundle, the generator automatically creates a virtual directory for the generated web application on the local host. By default the name of the virtual directory is [ApplicationName]_[BundleName]. This parameter allows assignment of a new name. The name is changed when next generating the bundle.

Visual Studio Version

This option specifies whether to use 'VS2005', 'VS2008' or 'VS2010' when generating a bundle.

The generator will automatically copy infrastructure files appropriate to the chosen value to the bundle views directory. This includes files such as project files for the generated solution.

X Scaling Factor

This specifies a scaling factor for increasing/decreasing the left position and width of controls.

Y Scaling Factor

This specifies a scaling factor for increasing/decreasing the top position and height of controls.

Other options are available specific to individual controls. Refer to the **CTC ASP.NET WebForms Configurator** documentation for further details.

3.2 Generating CopyFrom As Table

As an example, when setting the options `CopyFromAsTable` and `CopyFromAutoColumnHeaders`, a `CopyFrom` region painted as the following in the EAE or AB Suite development environment:

Most Recent Transactions:					
Ref	Date	Time	Tran	Vend/Cust	Quantity
IN-DOC	IN-DATE	IN-T	IN-IS	IN-VEN	IN-+
IN-DOC	IN-DATE	IN-T	IN-IS	IN-VEN	IN-+
IN-DOC	IN-DATE	IN-T	IN-IS	IN-VEN	IN-+
IN-DOC	IN-DATE	IN-T	IN-IS	IN-VEN	IN-+
IN-DOC	IN-DATE	IN-T	IN-IS	IN-VEN	IN-+
IN-DOC	IN-DATE	IN-T	IN-IS	IN-VEN	IN-+
IN-DOC	IN-DATE	IN-T	IN-IS	IN-VEN	IN-+

is generated as follows:

Ref	Date	Time	Tran	Vend/Cust	Quantity
3	09AUG08	1134	SALE	C2	3-
2	06AUG08	1516	SALE	C1	5-
7	06AUG08	1513	INGDS	V2	900+
1	06AUG08	1509	INGDS	V1	800+
1	06AUG08	1420	SALE	C1	

The look and feel of the table above is as generated by default. Each of the table controls (Table, TableHeaderRow, TableHeaderCell, TableRow and TableCell) are standard ASP.NET controls and can be styled using the CTC Configurator to suit local requirement.

A CopyFrom Table with a large number of copies occupying a large area on the form and causing the form to scroll can be placed inside a panel in order to limit the size and to provide scroll bars. This can be achieved using the CTC Configurator and specifying the table within an ASP.NET Panel control.

Alternating background color, as shown above, can be configured for the TableRow and TableCell controls. When alternating background color is specified for both the TableRow and TableCell a chequered effect is achieved.

When using the CopyFromAutoColumnHeaders option, the generator looks for Label controls painted directly above the table and places them as column headers inside cells in the TableHeaderRow. In order for Label controls to be discovered as column headers, they must be painted directly on the form. It is also assumed that a label has been painted for each column/field in the CopyFrom, including columns containing hidden fields, and columns containing multiple fields. It may therefore be necessary to add dummy (empty) label controls to the painted forms. When using AB Suite and label controls are painted inside a panel, they will not be discovered. The label controls are placed in the TableHeaderCells in the order they are painted.

In case the CopyFromAutoColumnHeaders option fails to produce a satisfactory result, the following two alternatives are available:

1. Label controls that are painted as column headers can be identified to the generator for the purpose of being used as column headers. As part of the label text, a column identifier can be specified like this: "Date[02]". The generator will recognize the notation [02] and use this as the column number. This requires the 'LabelIdDetection' option to be turned on.
2. Using the CTC Configurator, the individual TableHeaderCells can be configured to specify the text of the cell.

The number of columns created for a CopyFrom grid spanning multiple lines is determined by the number of controls painted on the first line. Controls painted on subsequent lines of a multiline CopyFrom region are positioned within the nearest column matching the position of the control.

Additional controls that are painted within a CopyFrom region, which are not associated with data fields, such as rectangles, lines and label controls, are not identified as CopyFrom

controls by the generate environment and will be removed from the form as long as they are positioned within the area directly occupied by the CopyFrom. Controls painted for visual effect, such as rectangles and lines, that cannot be identified as being within the CopyFrom are not removed.

When a CopyFrom region is painted inside a panel using AB Suite, all of the CopyFrom controls/fields must be painted directly on that panel. I.e. individual CopyFrom controls/fields must not be painted inside panels of their own.

3.3 Runtime Configuration

Parameters for the runtime configuration of the generated application can be specified via the CTC Configurator. When the runtime configuration parameters are changed, the generator will update the <appSettings> section of the Web.config file. This provides a convenient way of maintaining all configuration details in one place.

See the **CTC ASP.NET WebForms Configurator** documentation for further details.

3.4 Control Specifications

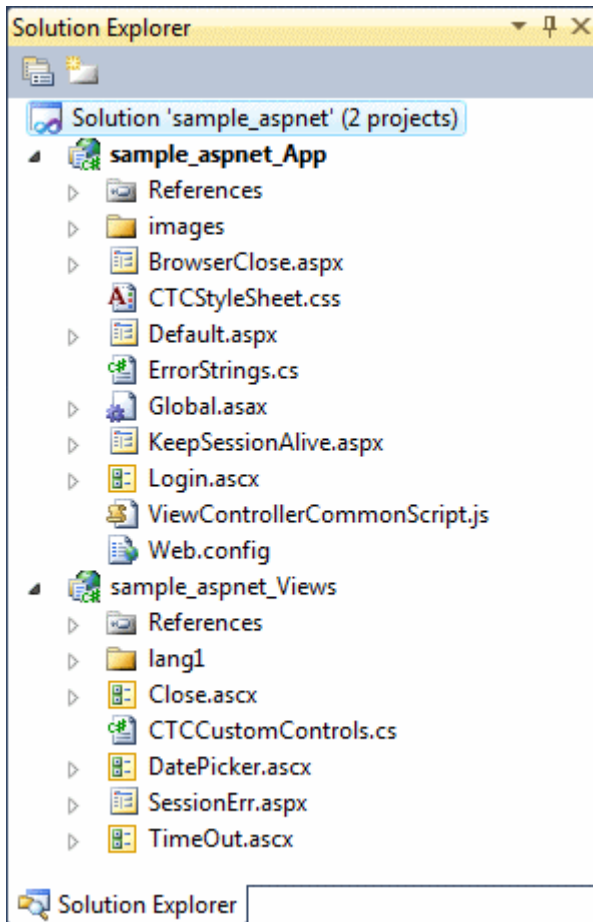
All properties on any Standard or Custom Control can be configured. The visual appearance of any control can be specified via the CTC Configurator. Any property defined by the Microsoft ASP.NET controls can be specified on the controls.

See the **CTC ASP.NET WebForms Configurator** documentation for further details.

4 The Generated User Interface Application

4.1 The Visual Studio Solution

The generated application is an ASP.NET WebForm application created for Visual Studio. A solution file containing two projects is created and the solution can be opened and inspected using Visual Studio. Below is an example of a generated solution.



The project named *sample_aspnet_App* contains fixed non-generated files required for the default ASP.NET Web Application. The Default.aspx file is the start-up page which is the host container for the CTC ASP.NET View Controller, which controls the display of the ispec forms as the user navigates through the application and manages all communication with EAE or AB Suite host system. For further information about the CTC ASP.NET View Controller, see section below.

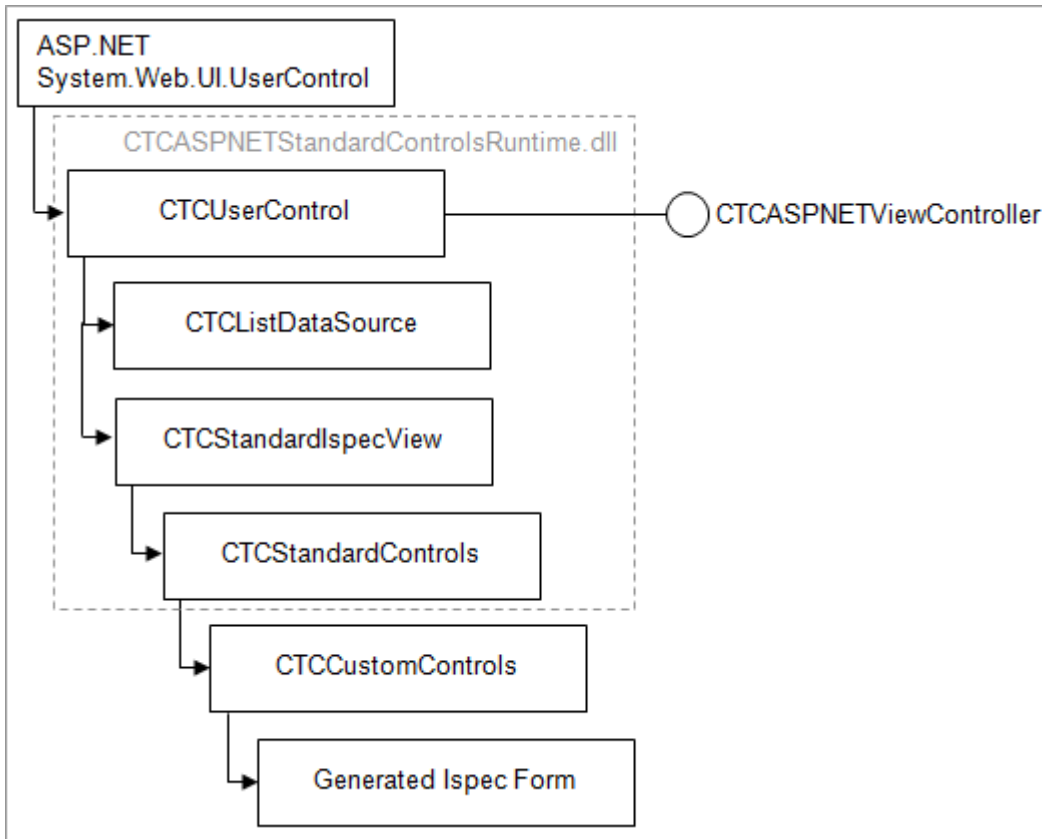
Files in this project can be customized and, because the project and these files are not generated, modifications are not overwritten by the generator.

The project named *sample_aspnet-Views* contains the generated forms located in the lang1-n directories. Only files in the lang1-n directories are generated and replaced for every generate process. All other files are fixed non-generated files and can be customized.

4.2 The Generated Ispec Forms

The form generated for each ispec in the bundle is created as an ASP.NET User Control. This allows the display of the forms as well as the communication to the host application to be managed by the CTC ASP.NET View Controller when running the forms.

In order to keep the generated code required for a form to a minimum and to provide a high level of flexibility, each form inherits its runtime behaviour from supplied infrastructure files. The inheritance hierarchy is as illustrated in the following diagram.



A form inherits from the CTCCustomControls class. This class implements functionality required for the runtime behaviour of any custom controls. This class is supplied as a source file named CTCCustomControls.cs, part of the generated project as seen in the Visual Studio illustration above. Initially it contains an example of how to implement a custom control. CTCCustomControls inherit from CTCStandardControls.

The CTCStandardControls class implements the runtime behaviour of all CTC Standard Controls. It inherits from the CTCStandardIspecView class, which provides the Ispec specific behaviour such as sending/receiving Ispec Models, which contain the data fields for an ispec, to/from the host system.

The CTCListDataSource class manages list data and provides the data binding for the ListDataSource control to lists returned from the host system.

The CTCUserControl class provides the interface to the CTCASPNETViewController and by inheriting from the Microsoft ASP.NET System.Web.UI.UserControl class it provides the Web User Interface behaviour. The CTCUserControl, CTCListDataSource, CTCStandardIspecView and CTCStandardControls classes are all supplied as a dll named CTCASPNETStandardControlsRuntime.dll which is installed in the bin directory of the generate solution.

5 CTC ASP.NET View Controller

The CTC ASP.NET View Controller is a generic ASP.NET control that manages all communication to EAE and AB Suite back end host systems. It can be included on any ASP.NET page on which access to an EAE/AB Suite host system is required. The Default.aspx delivered with the installation of the generator as described above, provides an example of how to include the View Controller in an ASP.NET application and shows how to use the View Controller.

The CTC ASP.NET View Controller manages the session with the host system starting with connecting to the host system, displaying ispec forms/views as the user navigates through the host application, sending and receiving data to and from the host system and ending with disconnecting from the host system. It uses the standard Component Enabler from Unisys to communicate with the EAE/AB Suite host system.

The CTC ASP.NET View Controller is configured using the standard Web.config. Configuration parameters can be set directly by editing the <appSetting> section of the Web.config file or by using the CTC Configurator and adding a 'runtimeConfiguration' element to the configuration of the bundle. For information about runtime configuration parameters see the **CTC Silverlight Client Configurator** document.

5.1 CTC ASP.NET View Controller API

The ASP.NET page (default.aspx) hosting the View Controller can control the behaviour of the View Controller by using the programmatic interface. The default.aspx page included with the installation of the CTC ASP.NET Generator provides an example as a starting point for customizing the ASP.NET User Interface application to suit local requirements.

5.1.1 Events

The ASP.NET main page can get control at key events during the end user interaction with the forms/views. The View Controller raises the following events:

- PreTransaction – Occurs after the end user submits the form and before the transaction is sent to the host system. This provides the opportunity for the application hosting the View Controller to check the data before it is sent to the host system and to bypass the transaction or to cancel further rendering of forms/views. Setting the BypassTransaction property on the event arguments causes the transaction to be ignored and not sent to the host system. Setting the CancelViewRendering property on the event argument causes the view to close and no further views will be shown until re-activated by the hosting application by calling the DisplayIspec method.
- PostTransaction – Occurs after the host system has responded to the transaction. This provides the opportunity for the application to check the data before it is presented to the user or to cancel further rendering of forms/views. Setting the CancelViewRendering property on the event argument causes the view to close and no further views will be shown until re-activated by the hosting application by calling the DisplayIspec method.
- StatusLine – Occurs after a response to a transaction has been received from the host system. This provides the opportunity for the application to check the status of the transaction and to customize the error handling. Setting the BypassStatusHandling property on the event arguments indicates the application is providing its own error

handling and the default error handling provided by the View Controller will be bypassed.

- **AlternateForm** – Occurs when the end user navigates to a new form/view, just before loading the new form. This allows the application to specify an alternative form/view to load instead of the generated form/view. This allows painting an alternative form to the form painted in EAE/AB Suite using Visual Studio or other external forms painters.
- **SessionClosed** – Occurs when the session with the host system has been closed. This gives the application the opportunity to know the status of the session and to close down the application or allow the user to re-connect.

5.1.2 Methods

Frequently used methods available on the View Controller include the following:

- **AutoConnect** – Connects to the host system and establishes a session using run-time parameters specified in web.config.
- **DisplayIspec** – Displays the current ispec. This method is the default method for displaying ispecs that are returned by the host system in response to a transaction.
- **Transmit** – Transmits the current IspecView to the host system. This should only be used when there is a need to transmit an IspecView outside of the normal behaviour. The normal behaviour is to automatically transmit an IspecView to the host when the user clicks on a button or hits the Enter key.
- **SetCurrentIspec** – Sets the specified ispec name as the current Ispec. This should only be used when there is a need to set a different ispec as the current outside of the normal behaviour.

6 Custom Controls

Custom Controls are controls that extend a Standard Control to implement specific requirements or to take advantage of technologies such as AJAX.

Custom Controls are used for substituting Standard Control on the generated form when the User Interface requirements demand an interface that cannot be satisfied by the standard controls. Using the CTC Configurator, Standard Controls can be substituted with Custom Control (see the **CTC ASP.NET WebForms Configurator** documentation for further details).


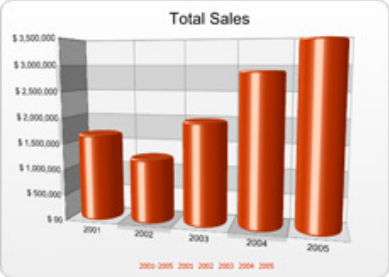
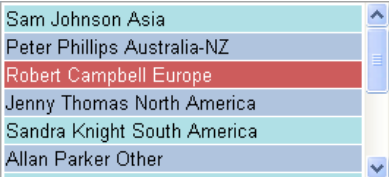
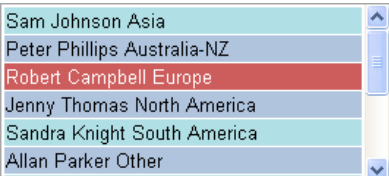
Included with the CTC ASP.NET WebForms Generator are a number of Custom Controls. These can be used out of the box or changed to suit local requirement.



6.1 System Provided Custom Controls

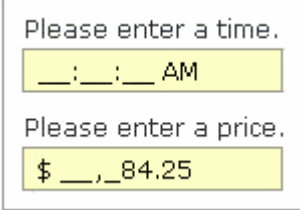
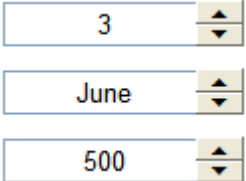
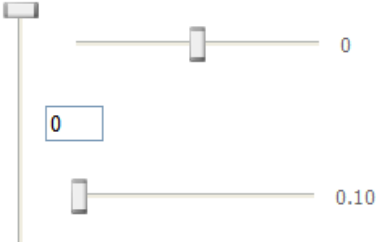
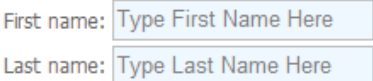
Below is a list of Custom Controls that are delivered as part of the CTC ASP.NET WebForms Generator.

Included in the list of custom controls are a number of controls from the Microsoft ASP.NET AJAX Control Toolkit. The ASP.NET AJAX Control Toolkit is a free download and contains more than 40 additional AJAX controls that work on top of the core ASP.NET AJAX release. Six controls from the toolkit that are particularly suitable for extending controls on EAE and

AB Suite forms to enhance the user experience have been included as custom controls. Other AJAX controls can be included on request. For more information on the ASP.NET AJAX Control Toolkit see <http://www.asp.net/ajax/ajaxcontroltoolkit/>. For information on downloading and installing the toolkit, see the CTC ASP.NET WebForms Generator ReadMe document.

Control	Description
<p>CalendarExtender-AJAX</p> 	<p>This extends the standard TextBox control where date input is required.</p> <p>It provides client-side date-picking functionality with customizable date format suitable for EAE and AB Suite and UI in a popup control. The user can interact with the calendar by clicking on a day to set the date, or the "Today" link to set the current date. Actions are provided that allow the user to easily jump to past or future dates within the calendar control.</p> <p>For further details see http://www.asp.net/AJAX/AjaxControlToolkit/Samples/Calendar/Calendar.aspx.</p> <p>This requires the ASP.NET AJAX Control Toolkit.</p>
<p>ComponentArt_WebChartControl</p> 	<p>This extends the standard ListBox control.</p> <p>This provides an example of utilizing a Third Party control from ComponentArt for displaying dynamic list data from EAE and AB Suite as a Chart.</p> <p>This requires the ComponentArt.Charting.WebChart.dll from Component Art.</p> <p>For further details on how to use the WebChart control from Component Art see: http://www.componentart.com/charting/.</p>
<p>DataListControl</p> 	<p>This extends the standard ListBox control.</p> <p>It takes advantage of the ASP.NET DataList control which provides capabilities for styling individual items in the list. It is especially useful for single column lists.</p>
<p>DataListControl-AJAX</p> 	<p>This extends the standard ListBox control.</p> <p>This is the same as the DataListControl but with added AJAX UI functionality that provides immediate feedback to the user when selecting an item in the list.</p> <p>This requires ASP.NET AJAX.</p>
<p>DatePicker</p>	<p>This extends the standard TextBox control where date input is required.</p>

	<p>This is based on the ASP.NET server-side Calendar control and provides a customizable popup calendar control for easy date selection in date formats suitable for EAE and AB Suite.</p>																		
<p>DatePicker-AJAX</p> 	<p>This extends the standard TextBox control where date input is required.</p> <p>This is the same as the DatePicker but with added AJAX UI functionality that provides immediate feedback to the user when selecting a date in the popup calendar control.</p> <p>This requires ASP.NET AJAX.</p>																		
<p>FilteredTextBoxExtender-AJAX</p> <p>Only digits are allowed here: <input type="text" value="120955"/></p> <p>Only lower-case letters are allowed here: <input type="text" value="filteredtextbox"/></p>	<p>This extends the standard TextBox control.</p> <p>FilteredTextBox prevents users from entering invalid characters into a TextBox.</p> <p>For further details see http://www.asp.net/AJAX/AjaxControlToolkit/Samples/FilteredTextBox/FilteredTextBox.aspx.</p> <p>This requires the ASP.NET AJAX Control Toolkit.</p>																		
<p>GridViewControl</p> <table border="1" data-bbox="236 1348 633 1527"> <thead> <tr> <th></th> <th>Sales Rep</th> <th>Geographic Area</th> </tr> </thead> <tbody> <tr> <td>Select</td> <td>Sam Johnson</td> <td>Asia</td> </tr> <tr> <td>Select</td> <td>Peter Phillips</td> <td>Australia-NZ</td> </tr> <tr> <td>Select</td> <td>Robert Campbell</td> <td>Europe</td> </tr> <tr> <td>Select</td> <td>Jenny Thomas</td> <td>North America</td> </tr> <tr> <td>Select</td> <td>Sandra Knight</td> <td>South America</td> </tr> </tbody> </table>		Sales Rep	Geographic Area	Select	Sam Johnson	Asia	Select	Peter Phillips	Australia-NZ	Select	Robert Campbell	Europe	Select	Jenny Thomas	North America	Select	Sandra Knight	South America	<p>This extends the standard ListBox control.</p> <p>It takes advantage of the ASP.NET GridView control which provides capabilities for styling individual items in the list including column headers. It is particularly useful for multi column lists.</p>
	Sales Rep	Geographic Area																	
Select	Sam Johnson	Asia																	
Select	Peter Phillips	Australia-NZ																	
Select	Robert Campbell	Europe																	
Select	Jenny Thomas	North America																	
Select	Sandra Knight	South America																	
<p>GridViewControl-AJAX</p> <table border="1" data-bbox="236 1594 633 1774"> <thead> <tr> <th></th> <th>Sales Rep</th> <th>Geographic Area</th> </tr> </thead> <tbody> <tr> <td>Select</td> <td>Sam Johnson</td> <td>Asia</td> </tr> <tr> <td>Select</td> <td>Peter Phillips</td> <td>Australia-NZ</td> </tr> <tr> <td>Select</td> <td>Robert Campbell</td> <td>Europe</td> </tr> <tr> <td>Select</td> <td>Jenny Thomas</td> <td>North America</td> </tr> <tr> <td>Select</td> <td>Sandra Knight</td> <td>South America</td> </tr> </tbody> </table>		Sales Rep	Geographic Area	Select	Sam Johnson	Asia	Select	Peter Phillips	Australia-NZ	Select	Robert Campbell	Europe	Select	Jenny Thomas	North America	Select	Sandra Knight	South America	<p>This extends the standard ListBox control.</p> <p>This is the same as the GridViewControl but with added AJAX UI functionality that provides immediate feedback to the user when selecting an item in the list.</p> <p>This requires ASP.NET AJAX.</p>
	Sales Rep	Geographic Area																	
Select	Sam Johnson	Asia																	
Select	Peter Phillips	Australia-NZ																	
Select	Robert Campbell	Europe																	
Select	Jenny Thomas	North America																	
Select	Sandra Knight	South America																	
<p>MaskedEditExtender-AJAX</p>	<p>This extends the standard TextBox control.</p> <p>MaskedEdit is an extender that attaches to a TextBox control to restrict the kind of text users can enter. MaskedEdit applies a customizable "mask" that permits only certain types of characters/text to be entered. The supported data formats are; Number, Date, Time and</p>																		

 <p>Please enter a time. <input type="text" value="__:__:__ AM"/> Please enter a price. <input type="text" value="\$ __, _84.25"/></p>	<p>DateTime.</p> <p>For further details see http://www.asp.net/AJAX/AjaxControlToolkit/Samples/MaskedEdit/MaskedEdit.aspx.</p> <p>This requires the ASP.NET AJAX Control Toolkit.</p>
<p>NumericUpDownExtender-AJAX</p> 	<p>This extends the standard TextBox control.</p> <p>NumericUpDown adds "up" and "down" buttons to a TextBox control that increment and decrement the value in the text box. The increment and decrement can be simple +1/-1 arithmetic or they can cycle through a provided list of values.</p> <p>For further details see http://www.asp.net/AJAX/AjaxControlToolkit/Samples/NumericUpDown/NumericUpDown.aspx.</p> <p>This requires the ASP.NET AJAX Control Toolkit.</p>
<p>SliderExtender-AJAX</p> 	<p>This extends the standard TextBox control.</p> <p>The Slider extender changes a TextBox control to a graphical slider that allows the user to choose a numeric value from a finite range. The Slider's orientation can be horizontal or vertical and it can act as a "discrete" slider, allowing only a specific number of values within its range.</p> <p>For further details see http://www.asp.net/AJAX/AjaxControlToolkit/Samples/Slider/Slider.aspx.</p> <p>This requires the ASP.NET AJAX Control Toolkit.</p>
<p>TextBoxWatermarkExtender-AJAX</p> 	<p>This extends the standard TextBox control.</p> <p>TextBoxWatermark provides "watermark" behaviour to a TextBox control. When a "watermarked" TextBox is empty, it displays a message to the user with a custom CSS style. Once the user has typed some text into the TextBox, the watermark goes away. This can be used to provide information to the user about the TextBox.</p> <p>For further details see http://www.asp.net/AJAX/AjaxControlToolkit/Samples/TextBoxWatermark/TextBoxWatermark.aspx.</p> <p>This requires the ASP.NET AJAX Control Toolkit.</p>

6.2 Creating Own Custom Controls

Custom Controls can be created to implement specific requirements when none of the Standard Controls cover the requirements.

Custom Controls can be used for extending ASP.NET controls or implementing third-party controls. Lots of third-party controls are available on the market and CTC Custom Controls capability makes it possible to include them in the generated forms.

Custom Controls can be implemented easily without the need to customize the whole generator. A Custom Control is created as a class using Visual Studio and when implemented, it is added to the generator using the CTC Configurator.

Once added to the generator, Custom Controls can then extend or substitute controls on the generated forms. Using the CTC Configurator, controls on the form can be configured to be substituted with Custom Controls and the condition for when to do the substitution. For further details, see the **CTC ASP.NET WebForms Configurator** documentation.

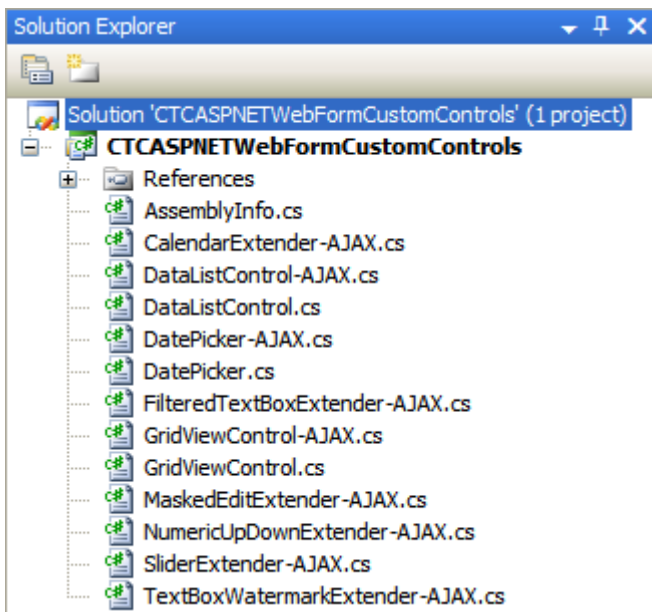
When creating Custom Controls, there are two distinct areas that need to be considered:

- The Generate Side
- The Runtime Side

6.2.1 Custom Controls – The Generate Side

Included with the installation of the CTC ASP.NET WebForms Generator is a sample Visual Studio project that provides 12 examples of how to implement the Generate Side of Custom Controls.

The project is named CTCASPNETWebFormCustomControls.csproj and is installed into the CustomControls directory of the [ceroot]\CTC-Software folder. The project is shown below.



When building the CTCASPNETWebFormCustomControls.csproj, a CTCASPNETWebFormCustomControls.dll is created and added to the bin directory of the [ceroot] folder. It may be necessary to modify the CTCASPNETWebFormCustomControls.csproj to point to the [ceroot] directory on the local machine.

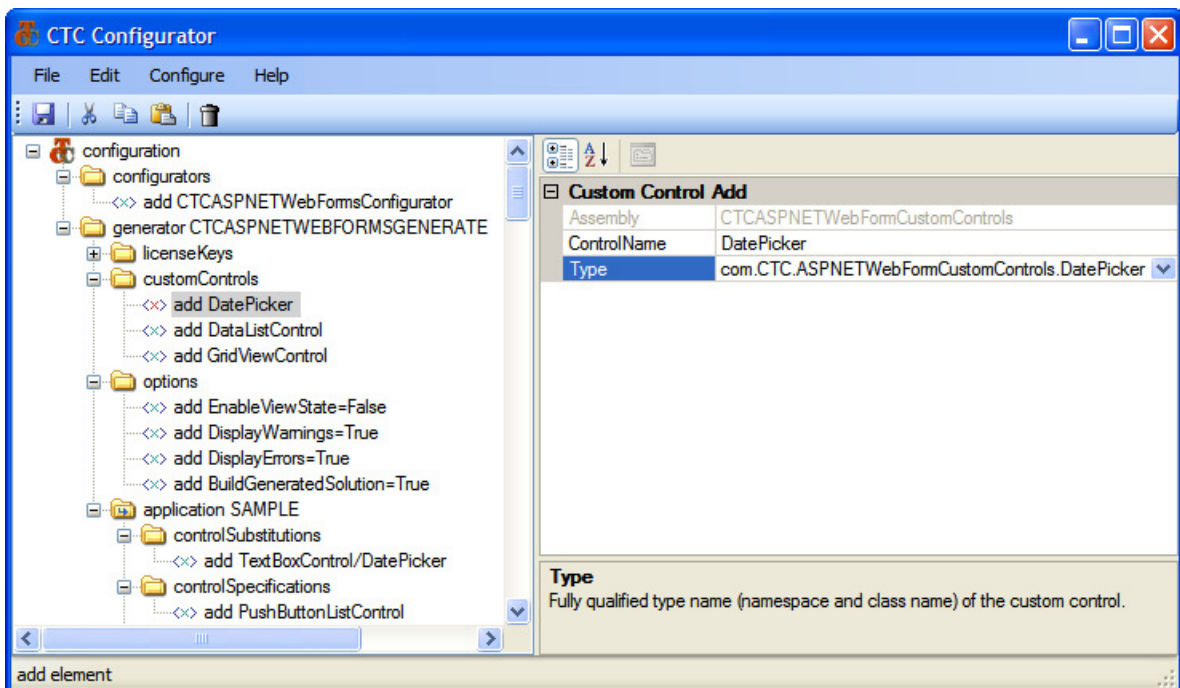
The DatePicker is an example of how to implement a composite ASP.NET control. The DatePicker class implements the Generate Side and extends the CTC Standard TextBox

Control to re-use as much of the generation of the TextBox control as possible. For the generation of a control, four methods need to be implemented:

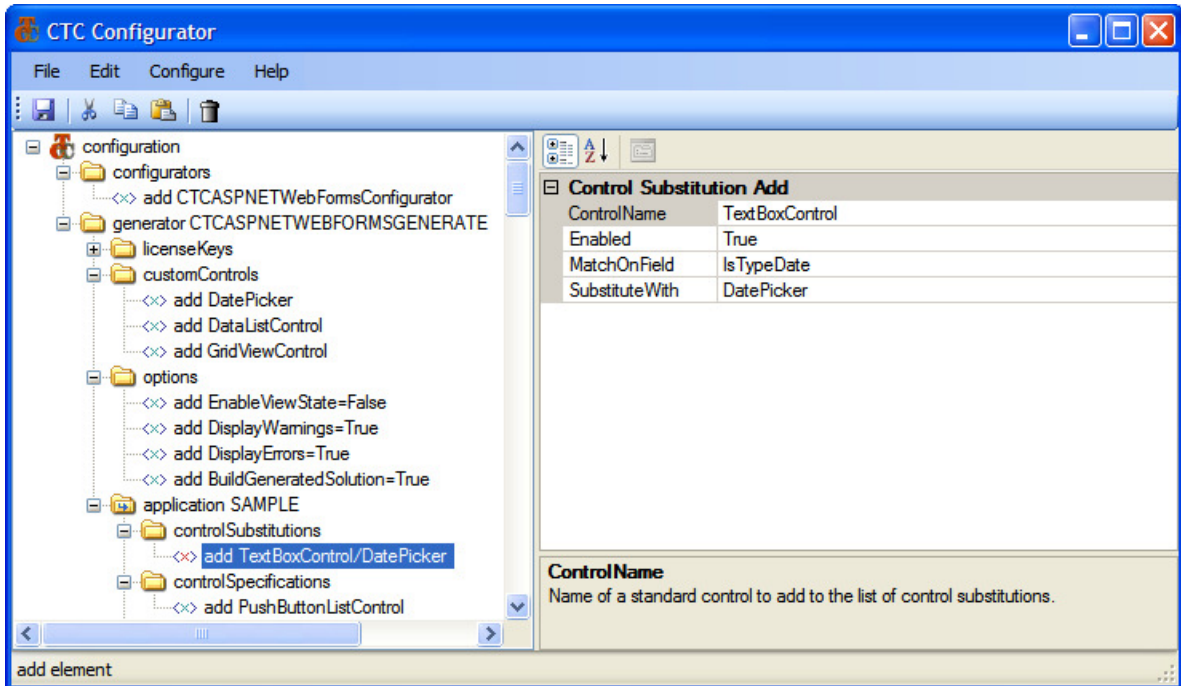
- `RenderControlSpecifications()`
This method outputs the specifications of the control to the .ascx file. These are the properties that define the look and feel.
- `RenderControlDeclarations()`
This method outputs the declaration of the control to the ascx.designer.cs file.
- `RenderCodeBehindSource_UpdateForm()`
This method outputs the code-behind that fills the form controls with values from the Ispec Model received from the host system. This is written to the UpdateForm() method of the ascx.cs file.
- `RenderCodeBehindSource-UpdateIspec()`
This method outputs the code-behind that fills the Ispec Model with values from the form controls before it is sent to the host system. This is written to the UpdateIspec() method of the ascx.cs file.

Included with the installation is also a DatePicker.ascx control. It provides the runtime side of the DatePicker and is an example of how to create an ASP.NET composite control. The DatePicker control generates the Specifications, Declaration, UpdateForm and UpdateIspec code required for including the DatePicker.ascx composite control into a form.

The following is an example of the DatePicker when added to the generator using the CTC Configurator.



Below is an example of the DatePicker when substituting the standard TextBox control with the DatePicker for date fields using the CTC Configurator.



The DataListControl and GridViewControl are two examples of how to utilise standard Microsoft ASP.NET controls such as DataList and GridView to replace the ListBox with controls that provide a much richer user experience. Both controls extend the CTC Standard ListBox Control and implement the four render methods as explained above.

6.2.2 Custom Controls - The Runtime Side

The runtime side of Custom Controls is implemented in the CTCCustomControls class, from which all generated forms inherit. The file CTCCustomControls.cs provides an example of how to implement the runtime side of the DataListControl and the GridViewControl.

Important methods to implement in the CTCCustomControls class are Get and Set methods for each control. The Get method returns the value of the control as it was entered or selected by the end-user in the browser. This value will be written to the corresponding field on the Ispec Model. The Set method sets the value from the corresponding field on the Ispec Model on the control to be shown to the end-user in the browser. The Get and Set methods are called from the UpdateIspec() method and UpdateForm() method respectively on the generated code-behind of the form.

Other methods and properties as required can be implemented on the class.