CTC Smart Client Generator

Version 1.0.5

Table of Contents

1 Introduction

1.1 What is CTC Smart Client Generator?

The CTC Smart Client Generator is a tool for creating a user interface for EAE 3.3 and AB Suite systems based on HTML and Javascript.

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 Smart Client 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 limited or no 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 Smart Client 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.

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 HTML Form Control. This allows the appearance of each control to be easily configured and, when necessary, Custom Controls can be created by extending the Standard Controls through inheritance. Custom Controls that map to other HTML 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 HTML 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 Smart Client 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 Smart Client Generator:

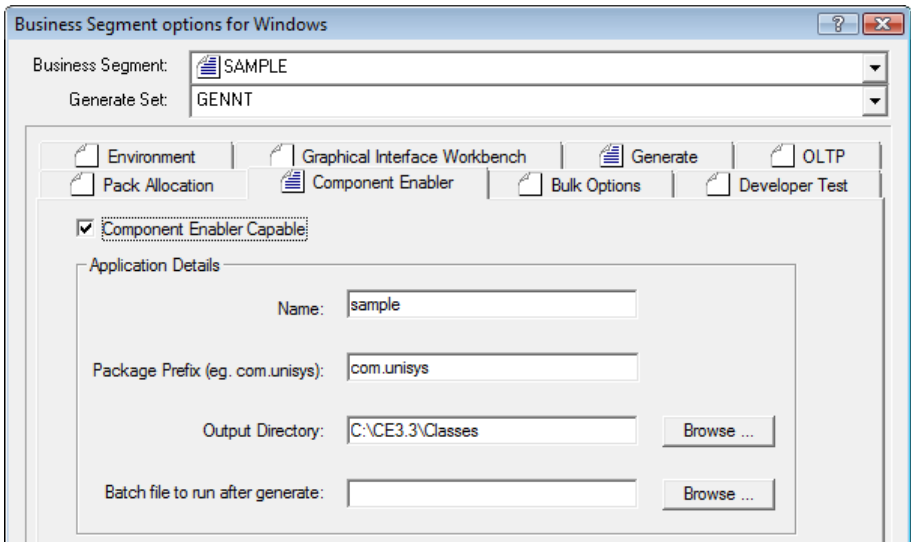
Control	Description
ButtonGroup	Container for a group of buttons such as Check Box, Push Button and Radio Button that is associated with the same field.
CheckBox	Check Box.
ComboBox	Drop down list box.
Form	Container for controls painted on a form.
Form Page	Outermost container for the form.
Grid	Grid container for CopyFrom regions.
GridHeaderRow	Grid Header Row container for CopyFrom regions.
GridHeaderCell	Grid Header Cell container for CopyFrom regions.
GridRow	Grid Row container for CopyFrom regions.
GridCell	Grid Cell container for CopyFrom regions.
Image	Image.
Label	Label.
Line	Horizontal, vertical and diagonal line.
ListBox	List Box.
Panel	Group container for other controls. AB Suite only.
PushButton	Push Button.
RadioButton	Radio Button.
Rectangle	Rectangular box.
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 and Password box.

2 Generator Initial Setup

The CTC Smart Client 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.

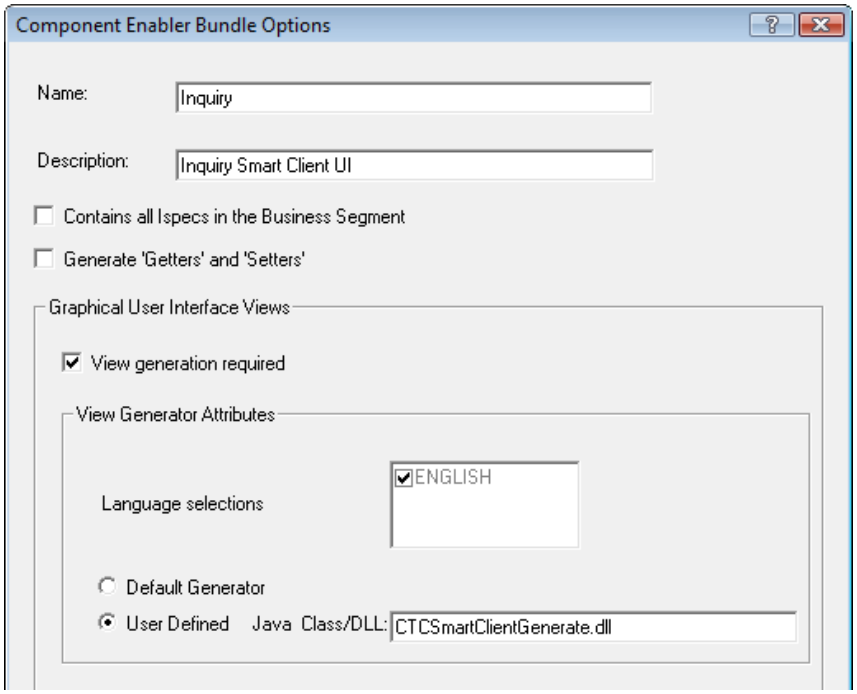
2.1 EAE Setup

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 Smart Client 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.

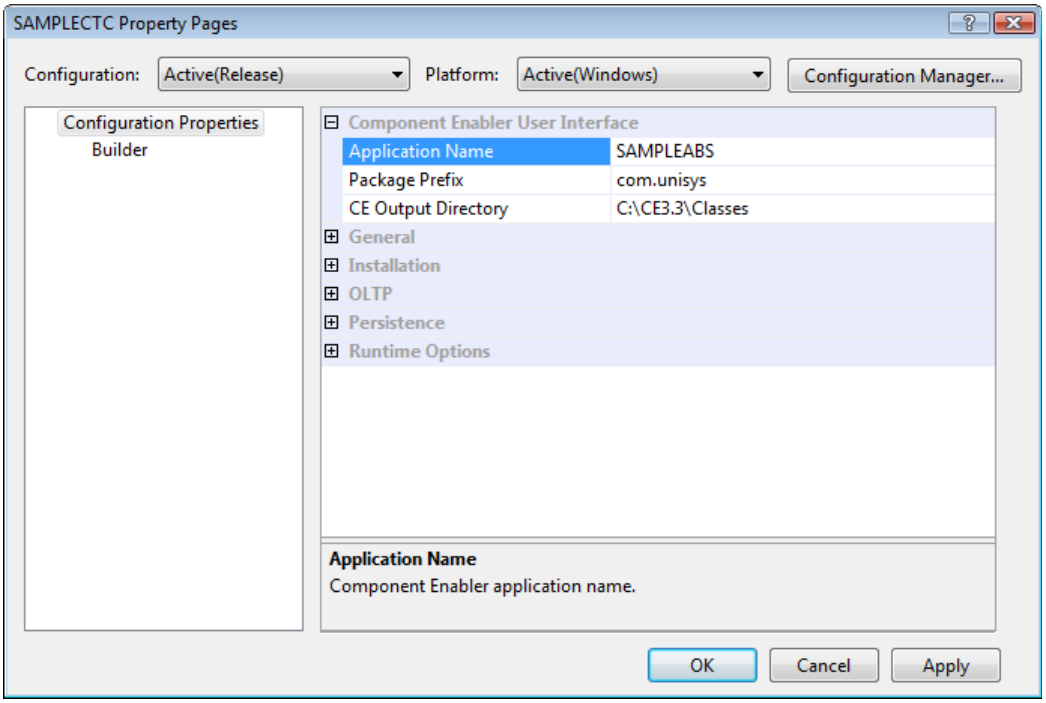
In addition to installing the CTC Smart Client Generator, users of EAE 3.3 IC3260 or earlier must also install the CTC Generate Gateway. The CTC Generate Gateway provides the necessary interface allowing the EAE Generate Environment to invoke generators implemented in .NET. Users of EAE 3.3 IC3270 or 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.

Ispec 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 'GenerateCSharpIspecModels=Y' to the Component Enabler section of the LINC.INI file. In addition it is recommended to also set GenerateJavaIspecModels=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.

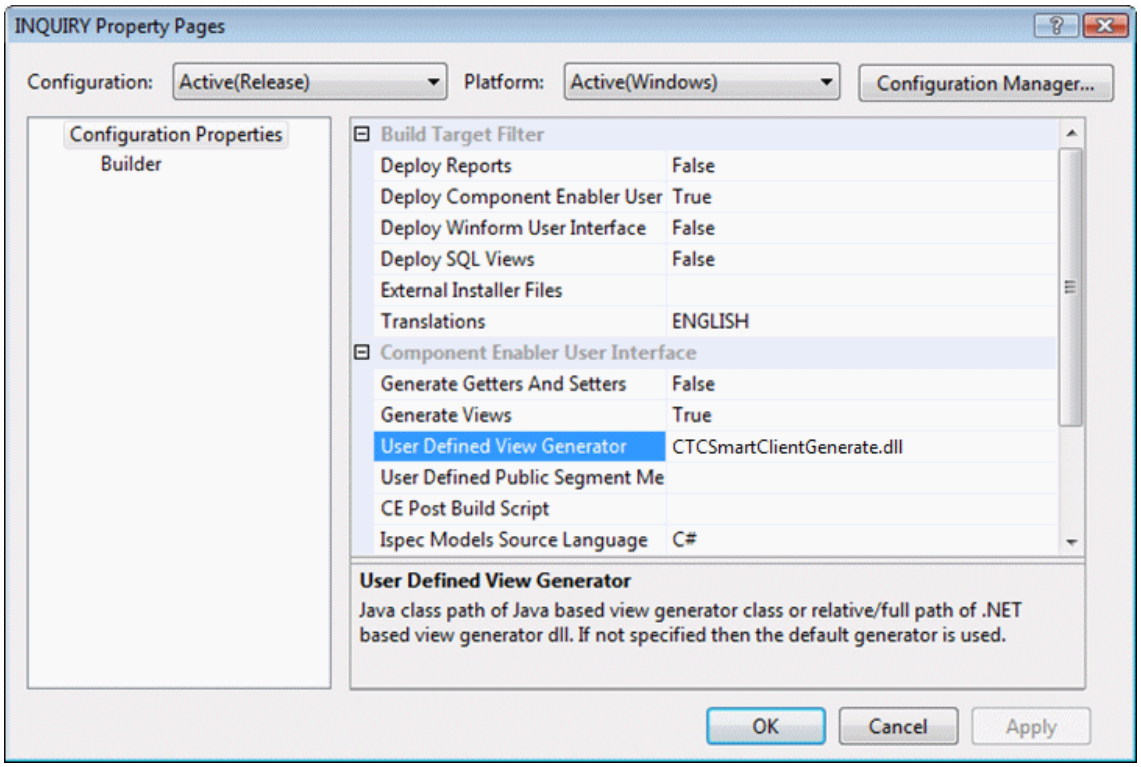
2.2 AB Suite Setup

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\sampleabs\inquiry.

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 Smart Client 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.

Ispec 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 for customization, 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 file are collectively referred to as Infrastructure Files.

No manual steps are required for copying the infrastructure files to the generate output directory. 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 'CTCSmartClientInfrastructureFiles.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]/ 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 Smart Client 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 Microsoft Expression Blend.

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.

Using alternate views provides the mechanism for form Designers and Developers to work together. Using tools like Microsoft Expression Blend allows Designers to paint forms taking advantage of all capabilities of HTML while Developers concentrate on the back-end implementation.

Alternate View Original Remove

When set and an Alternate View exists for an ispec, the original Ispec View will be removed from the generated project.

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.

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.

CheckBox RadioButton Custom Button

When true, enables custom button behaviour on Check Boxes and Radio Buttons.

CopyFrom As Grid

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

The grid is generated using the CTC Standard Controls for Grid, GridHeaderRow, GridHeaderCell, GridRow and GridCell. This provides the opportunity to configure and style the grid and all its elements taking advantage of all the properties available with HTML and CSS.

See the section 'Generating CopyFrom As Grid' 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 Grid.

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

CopyFrom Type

This option specifies the type of CopyFrom grid to generate.

The 'SimpleGrid' option generates the CopyFrom region as a row formatted grid.

The 'Grid' option generates the CopyFrom region as a column formatted grid. This is the default setting. See the section 'Generating CopyFrom As Grid' for further information.

CopyFrom Header Height

This option applies to CopyFrom Type SimpleGrid. It specifies the height above the copyfrom area where controls painted as headers will be moved into the SimpleGrid. Header controls must be positioned/painted within the left and right boundaries of the copyfrom area to be moved into the SimpleGrid as headers. The default value is 40px.

CopyFrom Left Margin

This option applies to CopyFrom Type SimpleGrid. It specifies the margin to the left of the copyfrom area where controls should be considered part of the copyfrom area. This would typically be non-field controls such as labels and rectangles. The default value is 40px.

Create Custom Metadata File

This option creates and initializes a CustomMetadata file with a list of all ispecs and fields in the bundle. The file is written to the [ApplicationName]_[BundleName]_Views folder.

The purpose of this file is to serve as a starting point for adding or modifying custom attributes for fields on ispecs. The file is an xml formatted file and can be modified with any text editor. The file is recreated on every generate, so it is advised to copy the file to another folder before modifying the file. This option is meant as a once off option and it is recommended to disable the option after the file has been created by the generator.

The CustomMetadata file contains a node for the bundle, however, this node can be removed to make ispecs and fields in the file applicable to the application regardless of the bundle.

When the option EnableCustomFieldMetadata is enabled, the generator read the CustomMetadata file and applies the attributes specified in the file to the fields on the ispecs. The generator reads the CustomMetadata file from the [CEROOT]\bin folder. The updated file therefore must be placed in the [CEROOT]\bin folder.

The content of the Metadata file is customisable using the CustomMetadata project provided with the installation of the generator. See also EnableCustomFieldMetadata.

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.

- afterInitializeView(): This method is invoked as part of initializing the view after the standard InitializeView() method is invoked. This method allows adding any additional view initializing code as required.
- afterOnLoaded():This method is invoked when the view has been added to the Visual Tree and after the standard onLoaded method has been called. This method allows adding any additional form loading code such as registering event handlers for the controls.
- afterHostResponse(): This method is invoked right after the host has responded to a transmit with an updated ispec model. The ViewModel and controls that

bind to properties on the ViewModel will have been updated before this method is invoked. The purpose of this method is to allow custom code to access controls on the form and fields on the ispec before the form is made available to the end user.

- afterListReceived(): This method is invoked right after the host has returned a list and the list data has been added to the IspecModel. For each transaction, this method is invoked once for each list that is received from the host. The purpose of this method is to allow custom code to access and inspect or change the list before it is accessible to the end user.
- beforeTransmitToHost(): This method is invoked right after the user submits the form and before the ispec model is sent to the host system. The purpose of this method is to allow custom code to access controls on the form and fields on the ispec model before it is sent 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 a valid extension.

Default Date Format

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

Possible formats are:

- UK (ddmmy, ddmmyy)
- US (mmddy, mmddyy)
- International (ymmdd, yymmdd)

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 ARIA

Enabling ARIA (Accessible Rich Internet Applications) requires running the generated application in browsers that support ARIA.

When enabling ARIA, the generator applies the following to the forms:

- The Help Text for EAE or the Tooltip Text for AB Suite will be added to Text Box controls and the input element of Combo Box controls as an aria-label property.
- An empty alt property will be added to image controls.

- A tab index with the value of 0 will be applied for Read-only field.

This may be enhanced in future releases depending on custom requirements.

Enable Custom Field Metadata

When enabled, the generator will call an external module (named CustomMetadata) for each field the option has been enabled for. The application name, bundle name, ispec name and field name will be passed into the module as parameters on the call. The module will return a list/collection of keyword/value pairs providing information for the generator to insert into the Control Template associated with the field.

Using the CTC Configurator, the user can modify the Control Templates and add keywords as required. The values of these keywords, which at generate time is provided via the external Custom Metadata module, will then be inserted by the generator.

As an example, by default, the CustomMetadata module will load the CTCSmartClientCustomMetadata.xml file located in the [CEROOT]\bin folder. This provides an example of how properties for fields can be maintained by the user in an external file. In this file, the user can add ispecs, fields and properties as required.

A sample Custom Metadata project is provided with the installation of the generator. For further information about customizing the Custom Metadata module, see the CustomMetadata project located in the [CEROOT]\CTC-Software\CTC Smart Client Generator\. Before customizing this project, copy the project to another location.

Exclude Language

When set, all ispecs and teach screens for the language will be excluded from the generate process. This option applies to the Language node only.

Exclude Teach Screen

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

Field AutoComplete

When set, previous entries for fields are stored and suggested as matches in an AutoComplete dropdown box.

It is recommended to set AutoComplete for selected alpha numeric fields rather than on all fields. Configuring AutoComplete for all fields can impact performance.

The system administrator determines which fields to apply autocomplete to. Each user can then determine whether they want to take advantage of this via the User Settings options dialog.

Field Change Highlight

When set and the 'Track Change' option is also set, the textbox element of the field will be highlighted when the user changes the value of the field. A css class 'ctc-dirty-field' is added to the text element when the value of the field changes.

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 Model files. This avoids errors occurring during the generation of the Ispec Model files caused by IIS locking Ispec Model files that have 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.

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 or DataGrid.

Log Level

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

Preserve White Space

When enabled, multiple space characters in labels are preserved.

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.

Radio Button Default First

When true, this option sets the first radio button in a group as selected when no default value has been specified for the field.

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.

Renew List From Host

When set to true, list data is retrieved from the host system on every transaction. Setting this option to false on lists that do not change during the course of transacting with the ispec avoids transferring the data on every transaction and improves the response time. The default value is true.

Submit On Enter Key

Allows submitting the form by hitting the enter key anywhere within the form.

TabIndex

This option specifies how to generate the TabIndex attribute on controls.

- 'AsPainted' create the TabIndex attribute with the value as specified when painting the form in EAE or AB Suite.
- 'Zero' create the TabIndex attribute with a value of zero on all controls. TabIndex 0 mean the tab order of controls is defined by the sequence the controls appear on the form. Controls are placed on the generated form in the order of Top to Botton and Left to Right. By specifying the DomOrderTop or DomOrderLeft attributes within the style property of the Control Specifications of a control, a control can be placed in a position within the HTML Document Object Model to suit the tab order within the natural flow.
- 'Resequence' create the TabIndex attribute with a sequential value in the order of Top to Bottom and Left to Right. The TabIndex value is created in multiples of 10.

TextBox Auto Select

Automatically highlight the text of a textbox when the textbox receives focus.

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 Character Casing

Specifies the character casing (Upper, Lower or Normal) to apply to a textbox for alphanumeric fields when typing into the textbox.

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 Numeric Only When Alpha

Ensures that only numeric digits can be entered into a textbox for fields defined as alpha.

TextBox Validate Numeric

Ensures that only numeric characters can be entered into a textbox for fields defined as numeric. This option validates the input character by character as the user enters the data. Data inserted in a field by copying/pasting is not validated. On some touch devices with a popup keyboard, such iPad, the shift key is not recognised. Therefore some special characters, which on a desktop keyboard are above the numeric keys, will not be rejected.

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.

Track Change

Enables tracking changes to fields, which may be used for alerting the user to save changes before moving to another record or ispec.

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.

Two Digit Year Cutoff

This option specifies an integer from 1 to 99 that represents the cutoff year for interpreting two-digit years as four-digit years. This option is used when presenting the two digit year of 6 digit date fields as a date including the century in controls such as the jQuery UI DatePicker.

A two-digit year that is less than or equal to the last two digits of the cutoff year is in the same century as the cutoff year. A two-digit year that is greater than the last two digits of the cutoff year is in the century that proceeds the cutoff year. For example, if two digit year cutoff is 56 (the default), the two-digit year 56 is interpreted as 2056 and the two-digit year 57 is interpreted as 1957. In other words, a two digit year cutoff of 2056 specifies dates in the 100 years range between 1957 and 2056. This is the equivalent of the Base Year specified on the Business Segment of EAE and AB Suite, which has a default value of 1957.

Virtual Directory Auto Create

This option configures the generator to automatically create a virtual directory for the generated Smart Client application. On machines without IIS, this option should be 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 the Visual Studio version to generate the bundle projects and solution for. The value can be VS2008 or VS2010.

When changing this property, infrastructure files appropriate for the selected option will be reinstalled next time when starting the generate of this bundle.

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 Smart Client Configurator** documentation for further details.

3.2 Generating CopyFrom As Column Formatted Grid

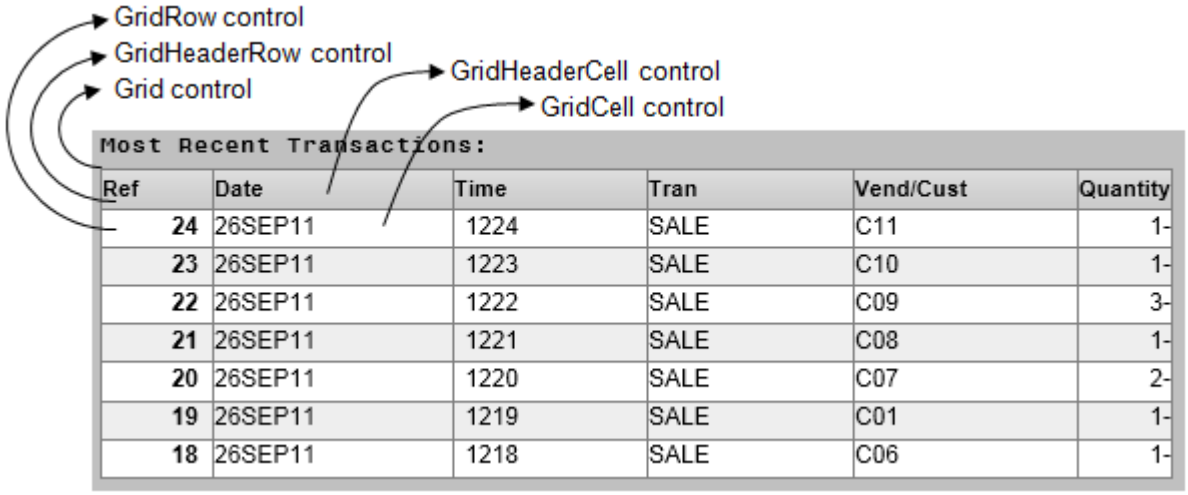
Generating a CopyFrom area as a Column Formatted Grid is suitable for copyfrom areas with field painted on a single line.

As an example, when setting the options CopyFromAsGrid, CopyFromAutoColumnHeaders and CopyFromType = Grid, 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:



Most Recent Transactions:

Ref	Date	Time	Tran	Vend/Cust	Quantity
24	26SEP11	1224	SALE	C11	1-
23	26SEP11	1223	SALE	C10	1-
22	26SEP11	1222	SALE	C09	3-
21	26SEP11	1221	SALE	C08	1-
20	26SEP11	1220	SALE	C07	2-
19	26SEP11	1219	SALE	C01	1-
18	26SEP11	1218	SALE	C06	1-

The grid is generated as an HTML Table control. Each of the elements in the grid (Grid, GridHeaderRow, GridHeaderCell, GridRow and GridCell) are standard CTC controls and can be styled using the CTC Configurator to suit local requirement.

Alternating background color, as shown above, can be achieved by styling the HTML Table using CSS.

Generating CopyFrom as a Grid is a process that requires the generator to make a number of assumptions as EAE/AB Suite Developer only provided a limited set of information for the generator to work with. The generator determines the grid layout by analyzing how and where the controls are painted within the CopyFrom area. The outcome can vary from ispec to ispec depending on different factors. This process only works well for simple CopyFrom ispecs with rows containing fields only (i.e. no labels ad rectangles) painted on a single line, and one header label control for each column. CopyFrom ispecs with rows containing fields and other controls painter over multiple lines often does not produce a good result as multiline CopyFrom rows often does not display well in a column formatted Grid.

When using the CopyFromAutoColumnHeaders option, the generator looks for Label controls painted directly above the CopyFrom area and places them as column headers inside cells in the GridHeaderRow. 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 GridHeaderCells 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 as: "Date[02]". The generator will recognize the notation [02] and use this as the column number.

   Using the Label control option requires the 'LabelIdDetection' option to be turned on.

   Specifying column numbers on label controls overrides the automatic column header detection. Therefore, when using this option, all labels that are meant as column headers must be identified with a column number.

2. Using the CTC Configurator, the individual GridHeaderCells 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 Generating CopyFrom As Row Formatted Grid

Generating a CopyFrom area as a Row Formatted Grid is suitable for all copyfrom areas with field painted on a single line as well as fields painted over multiple lines.

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

Number	Name/Address	Credit Limit	Select
C08	Alicia Henrickson PO Box 4455	2000	X
C10	Bennett Trucks 10 Parker Street	4000	X
C07	Brenda Fraser Box 98	4000	X
C05	Graham Edwards PO Box 789	8000	X
C06	Harry White Box 43	6000	X
C15	Helen Miles GPO Box 1990	6000	X
C03	Jennifer Palmer GPO Box 33130	2000	X

is generated as follows:

Number	Name/Address	Credit Limit	Select
C08	Alicia Henrickson PO Box 4455	2000	X
C10	Bennett Trucks 10 Parker Street	4000	X
C07	Brenda Fraser Box 98	4000	X
C05	Graham Edwards PO Box 789	8000	X
C06	Harry White Box 43	6000	X
C15	Helen Miles GPO Box 1990	6000	X
C03	Jennifer Palmer GPO Box 33130	2000	X

Alternating background color, as shown above, can be achieved by styling the HTML Table using CSS.

Options for generating CopyFrom area as a Row Formatted grid:

- CopyFrom Type
    - Value SimpleGrid.
- CopyFrom Auto Column Headers
    - Labels and other controls painted within a specified height above the copyfrom area are moved to the header of the SimpleGrid.
    - Header controls to be moved must be painted/positioned within the left and right boundaries of the copyfrom area.
- CopyFrom Header Height
    - This specifies the height above the copyfrom area that is considered header area. Controls painted/positioned within this area will be moved to the header of the SimpleGrid.
- CopyFrom Left Margin
    - This specifies the margin to the left of the copyfrom area where controls are considered to be part of the copyfrom. Only data fields are identified by AB Suite Developer to be copyfrom fields, and this option helps identify non-field controls such as labels and rectangles that are painted slightly to the left of the copyfrom area.

3.4 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 Smart Client Configurator** documentation for further details.

3.5 Control Specifications

All properties on any Standard or Custom Control can be configured. The visual appearance of any control can be specified either via the CTC Configurator or by styling the generated HTML controls using CSS. Any property defined by the HTML controls can be specified on the controls.

See the **CTC Smart Client Configurator** documentation for further details.

3.5.1 Control Properties

In addition to the standard HTML properties on the controls, some controls include properties that are specific to the CTC Smart Client Generator.

3.5.1.1 ComboBox, ListBox and DataGrid – List.XmlFilePath

List.XmlFilePath property specifies the path to an external XML formatted file to load and display in the list.

When specified, the file will automatically be retrieved from the server.

The path to the file is relative to the virtual directory from where the application originates.

For example:

List.XmlFilePath="XmlLists/Countries.xml"
points to http://localhost/Sample_Inquiry/XmlLists/Countries.xml.

3.5.1.2 ComboBox, ListBox and DataGrid – List.XmlElementPath

List.XmlElementPath property specifies the path to the xml element within the file holding the data to display in the list.

The path must be specified relative to the root element.

For example:

List.XmlElementPath="country"
specifies the country element within the root.

The XmlElementPath can specify a simple expression for the path to the element.

For example:

List.XmlElementPath="country[@code='GB']/city"
specifies the city element within the country which has the code attribute equal to GB.

When a complex expression is required, the expression must be specified in the code behind of the form that includes the list control.

3.5.1.3 ComboBox, ListBox and DataGrid – List.XmlSaveToLocalStorage

List.XmlSaveToLocalStorage property specifies to save the xml file in the Browser Local storage on the client work station.

When specified, the file will automatically be stored in the Local Storage. On subsequent access to the file it will be read directly from the local storage. A new version of the file will be retrieved from the server when a new version of the application is deployed to the virtual directory.

3.5.1.4 ComboBox, ListBox and DataGrid – List.Key

List.Key property is used for specifying the column in the list that is the key.

The key column holds the data that is sent to the host system.

For xml files, the Key property specifies an attribute name or element name within the XmlElementPath element.

For example:

1) List.XmlElementPath=”country” List.Key=”code” or
   List.XmlElementPath=”country” List.Key=”@code”
   specifies the code attribute within the country element.
2) List.XmlElementPath=”country” List.Key=”/code”
   specifies the code element within the country element.

For lists retrieved from the host system, the key property specifies the column number of the list that is the key column.

For example:

List.Key=”1”
specifies column 1 of the list as the key column.

3.5.1.5 ComboBox, ListBox and DataGrid – List.Columns

List.Columns property is used for specifying the columns to include in the list and the data type of columns.

For xml files, the Columns property specifies the attribute names or element names within the XmlElementPath element.

For example:

1) List.XmlElementPath=”country” List.Columns=”%code %name” or
   List.XmlElementPath=”country” List.Columns=”%@code %@name”
   specifies the code attribute within the country element as column 1 and the name attribute within the country element as column 2.
2) List.XmlElementPath=”country” List.Columns=”%/code %/name”
   specifies the code element within the country element as column 1 and the name element within the country element as column 2.

For lists retrieved from the host system, the Columns property specifies the column numbers of the list.

For example:

1) List.Columns=”%1 %2 %3”
   specifies to include columns 1, 2 and 3 of the list.
2) List.Columns=”%1 %2(0-15,15-40,55-5) %3”
   specifies to include columns 1, 2 with three sub columns (StartIndex-Length) and 3 of the list.

The %-sign is used as the separator between column specifications.

Specifying sub columns provides a way to create a multi column list even when the host system returns a single column list without making any changes to the host system.

By default all columns are returned as string values. However, in special cases when using DataGrid Column Sorting and Charting Controls, it may be necessary to convert numeric text data as numeric values.

As an example, to identify the third column in a list as a 'Numeric' column, the List.Column property is specified as follows:

List.Columns="%1 %2 %3:Numeric" or
List.Columns="%1 %2(62-6:Numeric) %3:Numeric"

When converting a value to numeric, depending on the formatting applied by the host ispec, it may be necessary to specify special characters used in the formatting such the Decimal Point, Thousand Separator and Currency Sign. These are specified as attributes on the Numeric property as in the following example:

List.Columns="%1 %2 %3:Numeric{dp=\\,|ts=.|cs=#}" or
List.Columns="%1
%2(62-6:Numeric{dp=\\,|ts=.|cs=#})
%3:Numeric{dp=\\,|ts=.|cs=#}"

When these formatting characters are not specified, the following default values are used:

Decimal Point = .
Thousand Separator = ,
Currency Sign = $

For the purpose of sorting a column containing dates, a column can be identified as a date column with the format of the date. The column will then be reformatted/converted to an internal sortable date format. As an example, to identify the third column in a list as a 'Date' column, the List.Column property is specified as follows:

List.Columns="%1 %2 %3:Date{dateFormat=mmddy|shortYearCutoff=56}" or
List.Columns="%1
%2(69-8:Date{dateFormat=mmddy|shortYearCutoff=56})
%3:Date{dateFormat=mmddy|shortYearCutoff=56}"

When converting a value to a date may be necessary to specify the date format of the column. When dateFormat is not specified, this defaults to the value specified on the DefaultDateFormat option configured for the ispec. A dateFormat must be specified when dates in a column are in a format that is different to the DefaultDateFormat.

For valid date formats see the jQuery Date Picker:
http://api.jqueryui.com/datepicker/#utility-formatDate

The shortYearCutoff can also be specified. When shortYearCutoff is not specified, this defaults to the value specified on the TwoDigitYearCutoff option configured for the ispec.

3.5.1.6 ComboBox, ListBox and DataGrid – List.MultiColumns

List.MultiColumns property specifies whether to return a multi column list from the host as a multi column list or as a list with one column in which multiple columns are concatenated into one string.

The default value is false and multi columns are concatenated into one column.

The MultiColumns property has effect only on lists retrieved from the host system. Xml files are always returned as multi columns.

3.5.1.7 ComboBox, ListBox and DataGrid – List.AddBlankRow

List.AddBlankRow property specifies whether to add a blank row at the beginning of the list. Providing a blank row in a list allows the end user to clear a selection by selecting the blank row.

By default, a blank row is added for ComboBox lists and no blank line is added for ListBox lists.

3.5.1.8 ComboBox, ListBox and DataGrid – List.Type

List.Type specifies the type of the list to return. By default, the generic 'CTCSmartClient.List_Row' type is used. However, in special cases such as charting controls, it may be necessary to use a specialized type.

The 'CTCSmartClient.List_Row.js', which can be found in the CTCSmartExtendedRuntime project, is an example of that.

3.5.1.9 ComboBox, ListBox and DataGrid – List.DependentList

List.DependentList specifies the field name of a list whose content depends on the selection of this list.

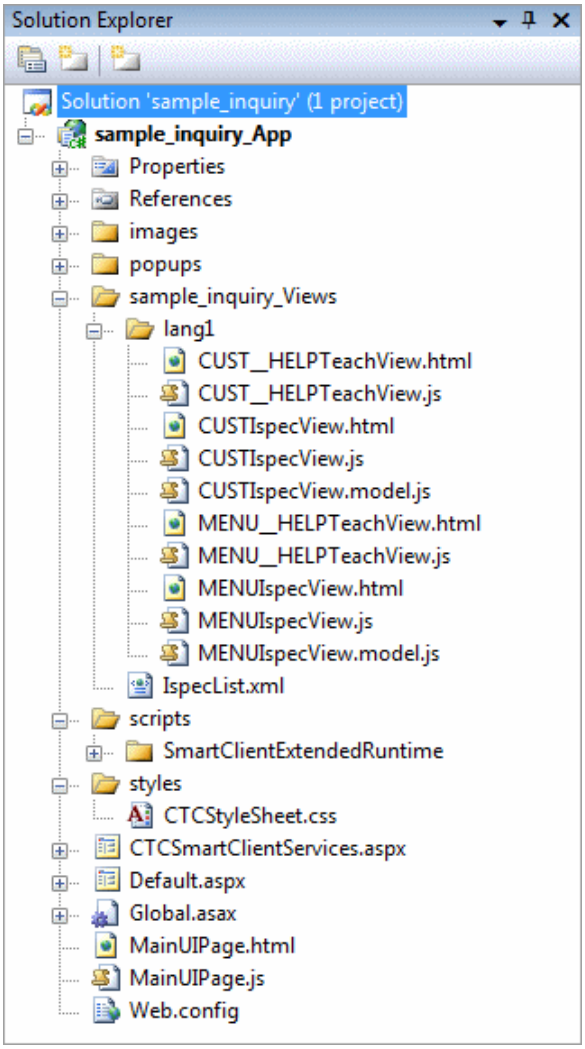
When a list field name is specified, appropriate code will be generated to automatically clear and renew the list content of the dependent list.

The actual content displayed in the dependent list depends on the expression specified in the List.XmlElementPath property of the dependent list.

4 The Generated User Interface Application

4.1 The Visual Studio Solution

The generated application is an HTML and Javascript application created for Visual Studio 2008 or 2010. A solution containing one generated project is created. The solution can be opened and customized using Visual Studio. To the left is an example of a generated solution with two ispecs, CUST and MENU, in the bundle.



The project named **sample_inquiry** represents the Smart Client application. It contains fixed non-generated files required for the Smart Client application on the client/browser side. The MainUIPage.html and MainPageUI.js files are the main page files that open in the browser when the user starts the application. These files contain the HTML and Javascript code behind of the Main application user interface. They host the CTC Smart Client 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 Smart Client View Controller, see section below.

The files CTCSmartClientServices.aspx and Default.aspx represent the server side of the application. They contain fixed non-generated files required for the application on the server side. The Default.aspx file is the start-up page for the Smart Client application. The URL to start the application points to this file.

CTCSmartClientViewControllerServices.aspx contains the Component Enabler services required by the client side to communicate with the host system.

The Web.config file contains the parameters to configure host system connectivity required by Component Enabler.

The folder named **popups** contains forms such as Login, SelectIspec, TransactionError, SplashScreen, CommandConsole, SystemMessages and Change Language.

The folder named **SmartClientExtendedRuntime** contains files that allow the customization of the runtime behaviour of the application by extending the default behaviour provided by CTC.

All files mentioned above are provided as source files allowing for the customization of the look and feel as well as the behaviour of the application to suit local requirements.

The folder named **sample_inquiry_Views** contains generated files for each of the ispecs in the bundle. These are downloaded to the client on demand as the user navigates through the application.

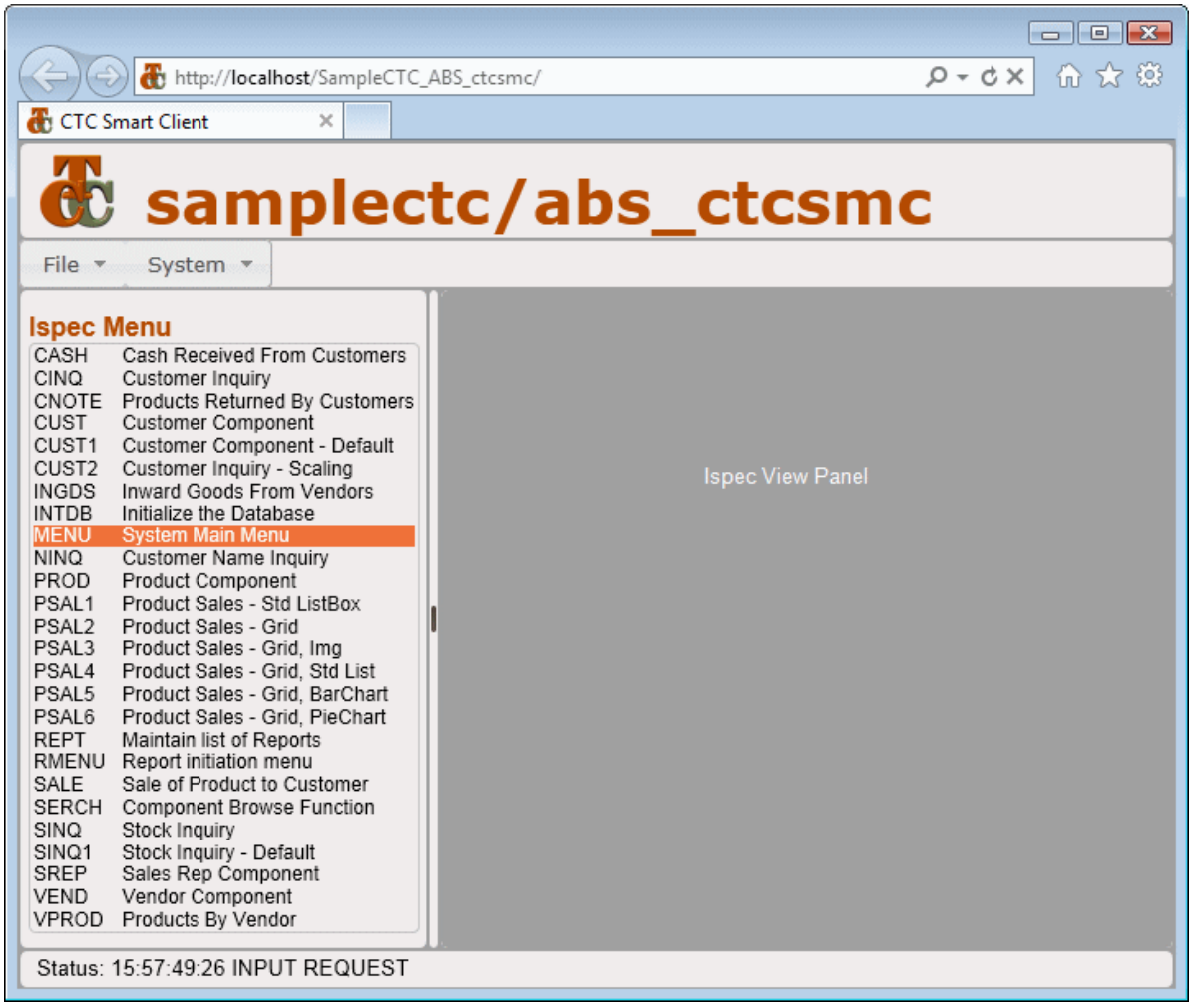
The CTC Smart Client Generator creates the following three elements for each ispec:

- TeachView - This is the teach/help information generated as an HTML form with Javascript code behind.
- IspecView – This is the ispec form containing all controls as they are painted at design time in EAE or AB Suite Developer generated as an HTML form including necessary Javascript code behind. Controls on the form bind to properties on the IspecView.model.
- IspecView.model – This is a Javascript module providing the interface between the controls on the form and the data held in the ispec model. This module contains one property for each data field defined on the ispec.

Ispecs are generated taking full advantage of the Knockout Javascript library data binding capabilities. Each control on the form uses the Knockout declarative data binding mechanism for displaying and editing data. There is no code being generated for binding to the data. This enables the complete separation of the presentation and the data, providing a flexible environment for customization if required.

4.2 Customizing Main User Interface Application

The Main User Interface application is the initial html page that is downloaded and displayed in the browser.



As seen in the image above, the Main User Interface application manages all elements of the user interface such as:

- The header
- The main menu bar
- The side menu
- The status line
- The ispec view panel

Delivered with the Smart Client Generator is a default Main User Interface application, which can be used as is or used as the starting point for customizing the User Interface application to suit specific local requirements.

The default User Interface application consists of the following two files located in the [app]_[bundle]_App folder of the views directory of the generated bundle:

- MainUIPage.html
- MainUIPage.js

As the first step, when customizing the User Interface application, it is recommended to make a copy of these files and customize the copied files. Follow the steps below to create a copy of the default User Interface application:

1. Open the generated solution in Visual Studio by double click on the **.sln** file located in the views directory of the generated bundle.
2. Copy and paste the **MainUIPage.html** file in the [app]_[bundle]_App project.
3. Change the name (e.g. to **MainUIPage_Custom.html**).
4. Copy and paste the **MainUIPage.js** file in the [app]_[bundle]_App project.
5. Change the name (e.g. to **MainUIPage_Custom.js**).
6. Open the MainUIPage_Custom.html file by double click on **MainUIPage_Custom.html**.
7. Locate **src="MainUIPage.js"** within MainUIPage_Custom.html and modify this to: src="**MainUIPage_Custom.js**"
8. Open the Default.aspx.cs file by double click on **Default.aspx.cs**.
9. Locate **MainUIPage** within Default.aspx.cs and modify this to: **MainUIPage_Custom**
10. Build the solution by right click on the Solution node and select **Build Solution**.

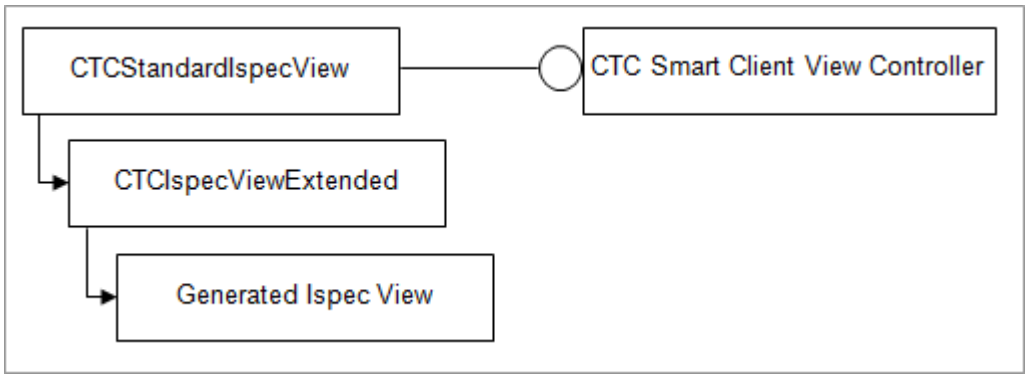
You can now start customizing the User Interface application by applying any modifications necessary to the custom copy of these files.

The custom copy of the Main User Interface application is run from the browser by entering the URL as normal (i.e. http://localhost/[ApplicationName]_[BundleName]/).

4.3 The Generated Ispec Forms/Views

The form/view generated for an ispec is created as a HTML Control.

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



A Generated Ispec View form inherits from the CTCIspecViewExtended class. This class inherits from the CTCStandardIspecView class and is a place holder for extending methods provided on the CTCStandardIspecView class. This class is supplied as a source file within the project named SmartClientExtendedRuntime, part of the generated project as seen in section 4.1.

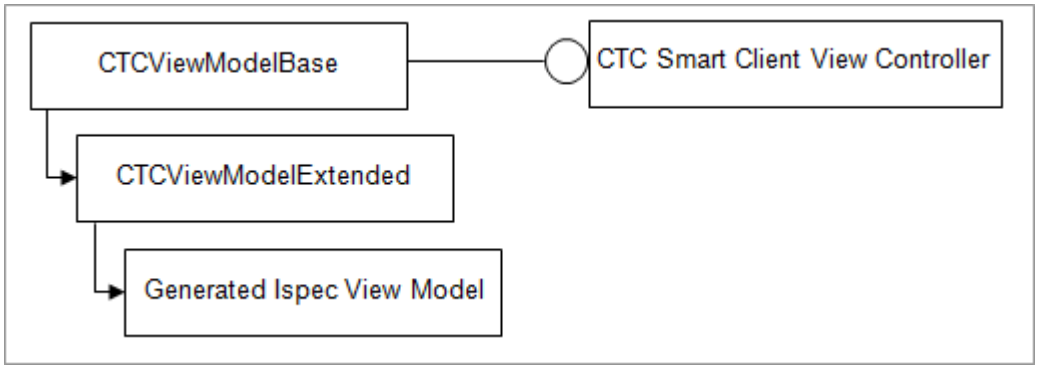
The CTCStandardIspecView class implements the runtime behaviour of all generated ispec views/forms such as Processing Dynamic Attributes, Button Focus, Auto Tabbing, Control Event Handling and Transmit to the Host System.

4.4 The Generated View Models

An Ispec View Model class is generated for each ispec. An Ispec View Model class represents the data model of the ispec. One property is generated for each field defined on the ispec. This provides the interface between controls on the form and the ispec data on the host system. The Ispec View Model class provides the ability for the controls on the actual View/Form to bind directly to the data without the need for any additional code behind.

This separation of the data from the presentation provides a flexible environment for customizing the forms. For example, for those users who have a requirement to go beyond the default generated forms, it allows a designer to create forms in a tool such as Expression Blend from Microsoft without having to be concerned about where and how to get the data.

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



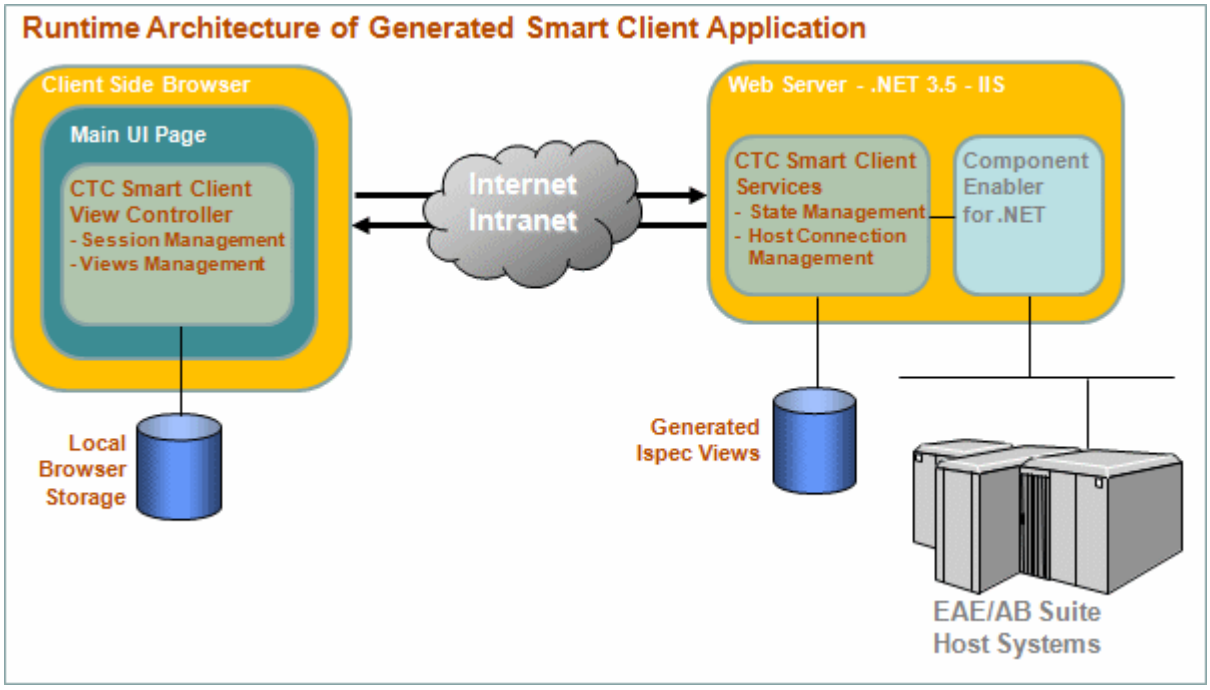
A Generated Ispec View Model inherits from the CTCViewModelExtended class. This class inherits from the CTCViewModelBase class and is a place holder for extending methods provided on the CTCViewModelBase class. This class is supplied as a source file within the

folder named SmartClientExtendedRuntime, part of the generated project as seen in section 4.1.

The CTCViewModelBase class implements various methods that are common for all generated Ispec View Models such as Radio Button List Converter, Checkbox Converter and Date Converter. In addition, the CTCViewModelBase class also provides the interface to the CTC Smart Client View Controller which provides access to the Ispec Model.

5 CTC Smart View Controller

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



The diagram above provides an overview of the runtime architecture of a Smart Client application generated by the CTC Smart Client Generator. It illustrates how the View Controller fits into the architecture as an essential component and shows the major functions it performs.

The CTC Smart Client 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. During the interaction with the host system, the View Controller calls upon services delivered by the CTC Smart Client Services module running on the web server. The session established with the web server does not timeout, as the View Controller automatically keeps it alive for the duration of the session.

The CTC Smart Client Services is implemented as an ASP.NET server component. It uses the standard Component Enabler from Unisys to communicate with the EAE/AB Suite host system. The state of a session established by the client View Controller is maintained throughout the session. This allows efficient exchange of data between the server and the

client. The amount of data required to be sent between the server and the client is kept to a minimum. By default, the state of connection to the host system is maintained throughout the session. However, the CTC Smart Client Services can be configured to use connection object pooling.

The CTC Smart Client Services are 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 Smart Client Configurator** document.

The CTC Smart Client View Controller includes a highly optimized Views Manager. Forms/views are downloaded individually from the web server as and when they are required, determined by how the user navigates through the application. When downloading forms/views, the Views Manager takes advantage of the Browser Local Storage facility for permanently storing the forms locally on the client. This enables the downloading of the forms/views once only until the application is regenerated.

5.1 CTC Smart Client View Controller API

The main user interface page hosting the View Controller can control the behaviour of the View Controller by using the programmatic interface. Included with the installation of the CTC Smart Client Generator is an example for utilizing the View Controller. The example is provided as a starting point for customizing the Smart Client User Interface application to suit local requirements. The examples are available in [ceroot]\CTC-Software\CTC Smart Client Generator\views\SmartClientApplication and include:

- MainUIPage (html + js): Main UI application installed as the default to the generated bundle.

5.1.1 Events

The 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.
- 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.
- 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.
- alternateView – 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 designers to use tools like the Microsoft Expression Blend for creating alternative forms that bind to the data properties of the generated View Models as mentioned above, and still take full advantage of the capabilities of the CTC Smart Client View Controller and the CTC solution.

- viewOpening – Occurs when a view/form is being opened as a result of calling the OpenIspec method. The MainUIPage must always listen to and handle the ViewOpening event. In this event handler the ispec view is added to the Visual Tree in the container where it is to be shown.
- viewLoad – Occurs when a View/Form has been loaded into the visual tree and before it is displayed to the user.
- ispecReturnedByHost – Occurs when the host system returns an ispec that is different to the current ispec as a response to a transmit/transaction. This would occur when the host ispec does a recall of another ispec.
- hostSessionClosed – 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.
- languageChanged – Occurs when the language has been changed by the host system.
- switchTo – Occurs when a SwitchTo has occurred on the host system.

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.
- openIspec – Opens or re-opens the specified ispec and returns an instance of the IspecView form. The caller displays the IspecView in a container on the user interface. This method is used for opening multiple ispecs within the same session and should only be used with 'stateless' EAE and AB Suite applications.
- closeIspec – Closes and removes an ispec from the collection of open ispecs.
- closeAllIspecs – Closes all ispecs and clears the collection of open ispecs.
- closeSession – Closes the current session with the host system.
- 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.

5.1.3 Properties

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

- currentIspecInfo – This provides information about the current active ispec such as the IspecModel, IspecView and IspecViewModel.
- currentIspecView – This points to the current active IspecView form.
- openIspecs – This holds a collection of current open ispecs.

5.2 Multiple Ispecs

The CTC Smart Client View Controller allows the User Interface to open and interact with multiple ispecs concurrently. Depending on whether the host applications are designed and implemented as 'state full' or 'stateless' systems, the CTC Smart Client View Controller provides two ways of achieving this.

A 'state full' system is a system that maintains the state of transactions on the host side. State information is typically stored in Global Work. Navigation between ispecs is often controlled by the host application and typically only one transaction at a time within a session is allowed. This means, if the user requires two ispecs open at the same time, two independent sessions are required. See "Multiple Sessions" below.

A 'stateless' system is a system that maintains the state of transactions using hidden fields, which are defined as part of the transaction. This means the state information follows the transaction. This allows the navigation between ispecs to be determined on the client side and provides the opportunity for the client to keep more than one transaction/ispec open at the same time within the same session. See "Multiple Concurrent Open Ispecs in the Same Session" below.

5.2.1 Multiple Concurrent Open Ispecs in the Same Session

The CTC Smart Client View Controller is designed to manage multiple open ispecs at the same time within the same session. This can be utilized by EAE and AB Suite applications that are 'stateless' and therefore allowing the state of open ispecs to be managed on the client side within the User Interface application.

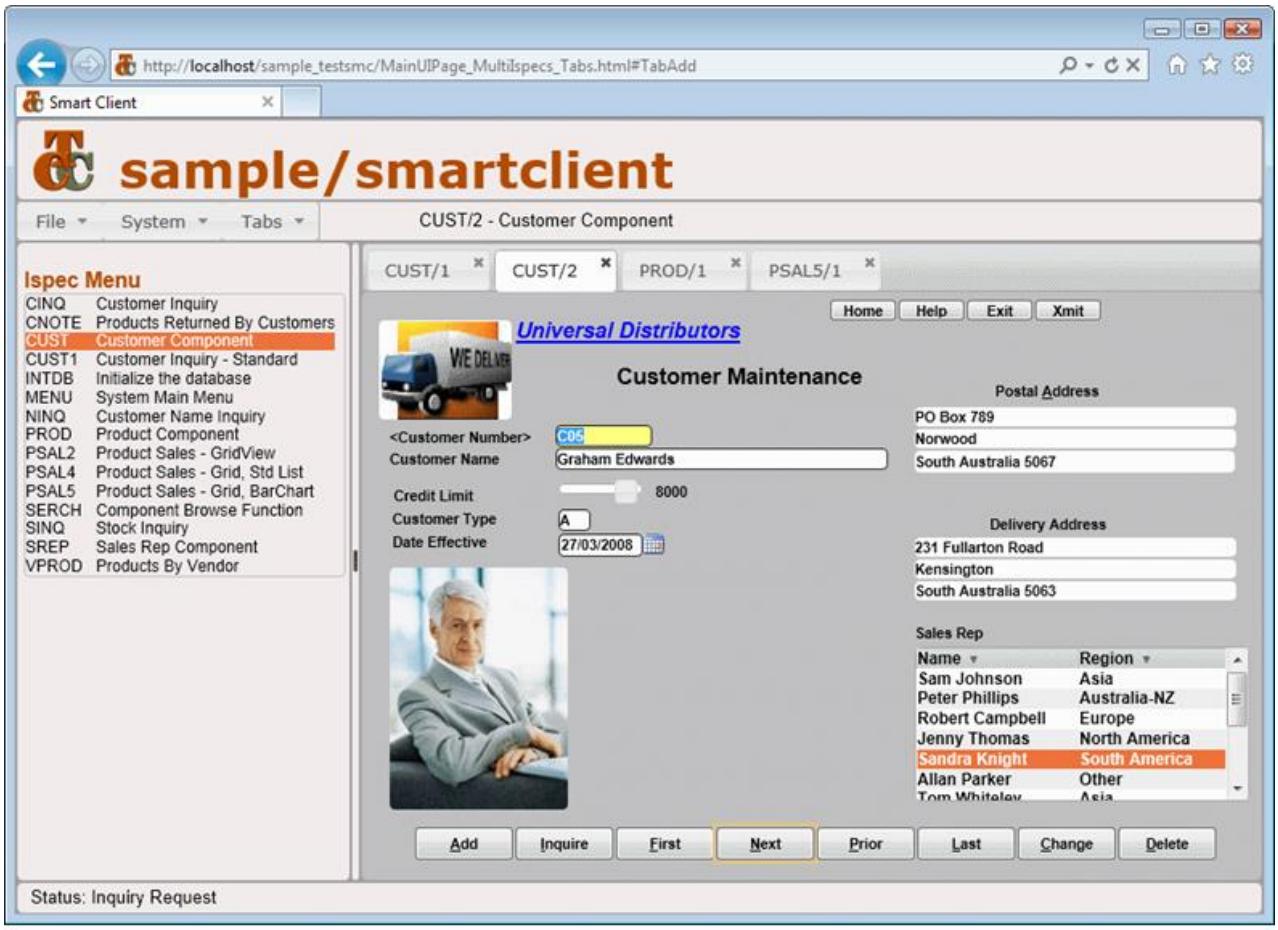
This allows the end user to open and close ispecs as necessary and navigate between open ispecs. Ispecs can be displayed in tabs or in windows. Any number of ispecs can be open at the same time. This also allows for the opening of multiple copies of the same ispec.

Opening and closing ispecs and keeping track of the state of each of the open ispecs is completely managed on the client side, making it very efficient. When the end user navigates between already open ispecs, no messages are sent to the host system. Only when the end user submits an ispec is a message sent to the host system.

5.2.1.1 Ispecs Displayed in Tabs

Below is an example that shows 4 ispecs open at the same time each being displayed in a separate tab.

When a new ispec is requested from the host, it is displayed in the current active tab. From the menu, the user can add new tabs in which to display ispecs. When selecting a tab, the ispec within the tab is made the current active ispec allowing the user to transact with the ispec.



Multiple Ispecs in Separate Tabs

Based on the 'standard' Sample system, delivered with the installation of the CTC Smart Client Generator, is an example of a Smart Client User Interface application utilizing this capability. The example is installed in the following folder:

"[ceroot]\CTC-Software\CTC Smart Client Generator\views\SmartClientApplication"

- MainUIPage_MultiIspecs_Tabs – This shows multiple open ispecs within the same session displayed in tabs.

This example can be used as the basis for customization to suit local requirements.

To install this example into the view directory of generated bundle follow the steps below:

1) Copy **MainUIPage_MultiIspecs_Tabs.html** to the views\[app]_[bundle]_App\ folder of the generated bundle.
2) Copy **MainUIPage_MultiIspecs_Tabs.js** to the views\[app]_[bundle]_App\ folder of the generated bundle.
3) Open **views\[app]_[bundle]_App\default.aspx** in a text editor such as Visual Studio and change:
   startUpPage = "**MainUIPage**"
   to:
   startUpPage = "**MainUIPage_MultiIspecs_Tabs**"
4) Open the browser and enter the URL of the application as normal.

5.2.1.2 Ispecs Displayed in Windows

Below is an example that shows 4 ispecs open at the same time each being displayed in a separate window. The windows can be minimized and maximized as well as arranged tiled, horizontally or vertically.

When a new ispec is requested from the host, it is displayed in the current active window. From the menu, the user can add new windows in which to display ispecs. A list of open windows is shown in 'Window' sub-menu. When selecting a window, the ispec within the window is made the current active ispec allowing the user to transact with the ispec.

Multiple Ispecs in Separate Windows arranged Tiled

Based on the 'standard' Sample system, delivered with the installation of the CTC Smart Client Generator, is an example of a Smart Client User Interface application utilizing this capability. The example is installed in the following folder:

"[ceroot]\CTC-Software\CTC Smart Client Generator\views\SmartClientApplication"

- MainUIPage_MultiIspecs_Windows – This shows multiple open ispecs within the same session displayed in windows.

This example can be used as the basis for customization to suit local requirements.

To install this example into the view directory of generated bundle follow the steps below:

5) Copy **MainUIPage_MultiIspecs_Windows.html** to the views\[app]_[bundle]_App\ folder of the generated bundle.

6) Copy **MainUIPage_MultiIspecs_Windows.js** to the views\[app]_[bundle]_App\ folder of the generated bundle.

7) Open **views\[app]_[bundle]_App\default.aspx** in a text editor such as Visual Studio and change:
   startUpPage = "**MainUIPage**"

to:

startUpPage = "**MainUIPage_MultiIspecs_Windows**"

8) Open the browser and enter the URL of the application as normal.

5.2.2 Multiple Sessions

The CTC Smart Client View Controller is designed to run multiple sessions within the same User Interface application. This can be utilized by EAE and AB Suite applications that are 'state full' and therefore requiring a separate session for each ispec to be displayed concurrently. For each session that is required, a CTC Smart Client View Controller is added to the User Interface application.

Concurrent access to multiple ispecs can also be achieved by starting multiple browsers and opening a session for each browser.

If assistance is required for utilizing multiple CTC Smart Client View Controller within the same User Interface Application, contact CTC.

6 System Trace

When required, system trace can be turned on by the administrator by enabling the following parameters in the web.config file:

- LoggingEnabled – This writes the standard Component Enabler trace information to the log file.
- CTCSmartClientLoggingEnabled – This writes trace information related to the CTC Smart Client on the web server and the client to the log file.
- MultiSessionLogging – This creates a separate log file for each client.

Enabling logging centrally in the web.config file turns logging on for all users, hence this would only be appropriate in a test/development environment.

In addition, logging can also be turned on for individual client sessions at runtime. System trace can be turned on/off at any time during a client session by entering the sequence Ctrl+Shift+PageUp (on a touch device tap 4 times on the header logo). When turned on, log information is written to a separate log file on the web server specifically for this client session.

The log file can be retrieved from the web server and displayed on the client for analysis by entering the sequence Ctrl+Shift+PageDown (on a touch device tap 5 times on the header).

7 Deployment

Smart Client applications are deployed from a web server which requires selected directories and files from the generated application to be copied to the deployment web server. The following explains how to move an application to the web server for two different scenarios:

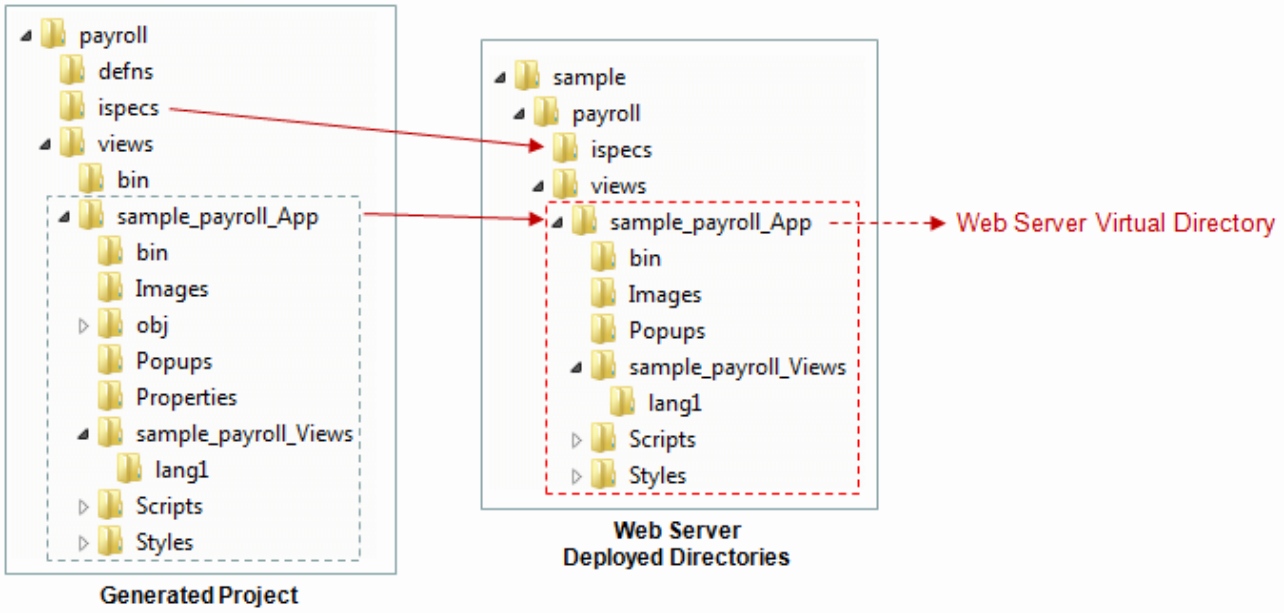
- Single Application
- Switch.To Application

7.1 Single Application

Deploying a single application is the typical case and also the most straight forward scenario. The diagram below shows the directories and files to be copied from the generated project to the web server:

- Ispec model files must be copied to the web server maintaining the same directory structure and folder names.
- Create a directory on the web server and configure this as the application Virtual Directory in IIS.
- The application_bundle_App folder including the sub folders must be copied to the Virtual directory as shown in the diagram below.

To assist with the copying, a bat file named DeploySolution.bat located in the bundle views directory is provided. This bat file can be modified to suit local requirements.



7.2 Switch.To Application

A Switch.To application is an application that consists of two or more generated application bundles, which the host system can switch between. As a Smart Client UI application is deployed from a web server, all files required for the application must be available within the same Virtual Directory. This means files from each of the generated application bundles must be copied to the web server and merged into the same Virtual Directory.

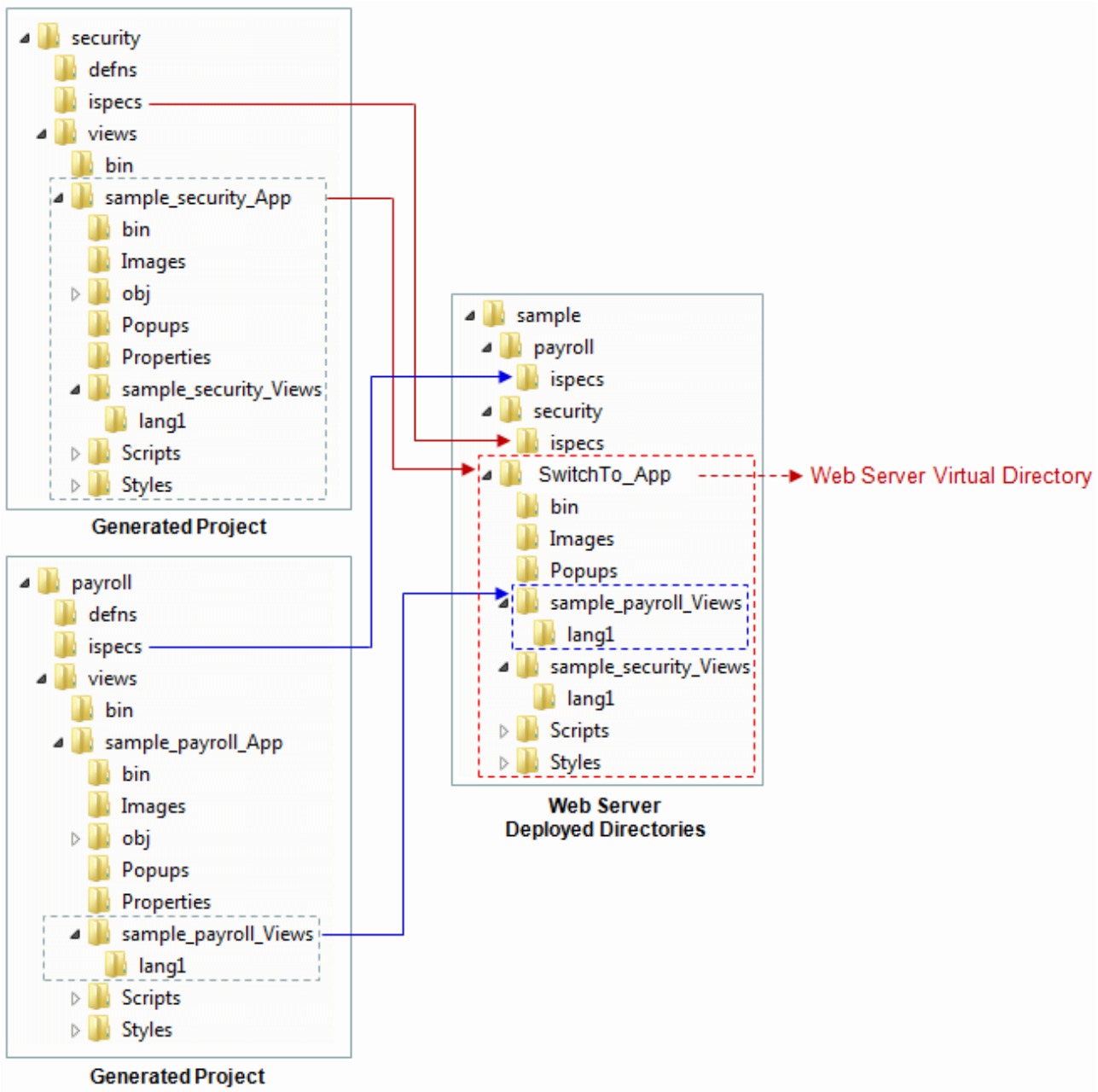
The diagram below illustrates the folders and files to be copied and merged from the generated projects to the web server:

- Copy Ispec model files for each of the applications to be part of the total Switch.To application to the web server maintaining the same directory structure and folder names.
- Create a directory on the web server and configure this as the application Virtual Directory in IIS. In the diagram below, SwitchTo_App is the Virtual Directory.
- Copy the application_bundle_App folder including sub folders of the main application to the Virtual Directory. As an example, in the diagram below, the security application

is the main application and the content of the sample_security_App folder is copied to the SwitchTo_App folder. This is the same as for deploying a single application.

- Copy the application_bundle_Views folder for each additional application to be part of the total Switch.To application to the Virtual Directory as shown in the diagram below. As an example, in the diagram, the sample_payroll_Views directory is merged into the SwitchTo_App folder.
- The folders Images, Popups, Scripts and Styles are shared between all applications involved in the total Switch.To application. Image, Popup, Script and Style files that are unique for an application must be copied into each of these folders within the Virtual Directory.

The ApplicationSwitching parameter in Web.config must be set to true in order to enable application switching.



8 Custom Controls

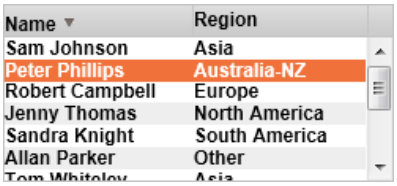
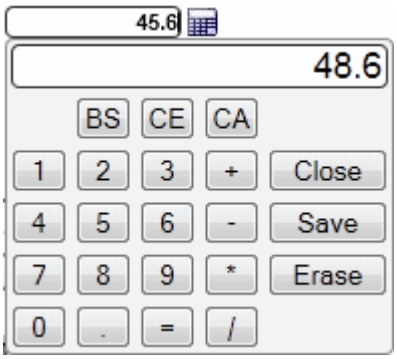
Custom Controls are controls that extend a Standard Control to implement specific requirements or to take advantage of third party controls.

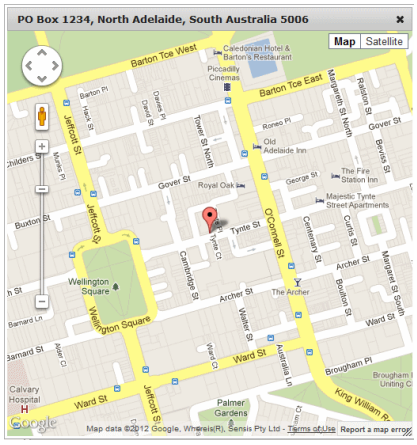
Custom Controls are used for substituting Standard Controls on the generated form when the User Interface requirements demand an interface that cannot be satisfied by the standard controls. Using the CTC Configurator, Standard Controls can be substituted with Custom Controls (see the **CTC Smart Client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

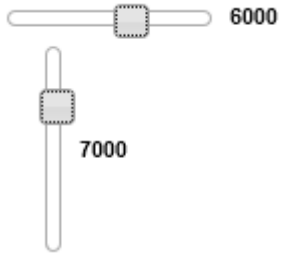
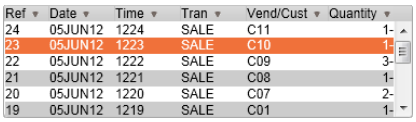
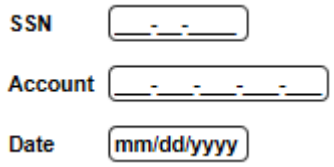
Included with the CTC Smart Client Generator are a number of Custom Controls. These can be used out of the box or changed to suit local requirement.

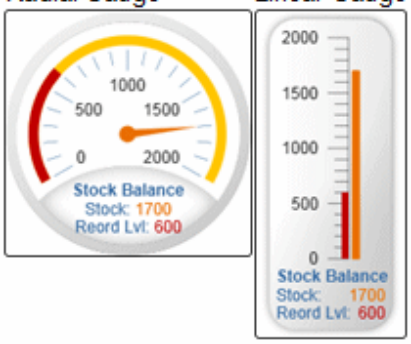
8.1 System Provided Custom Controls

Below is a list of Custom Controls that are delivered as part of the CTC Smart Client Generator. Additonal controls can be included on request.

Control	Description
DataGrid 	This extends the standard ListBox control. The Data Grid control is a flexible control with many options such as column headers, column sorting, and styling.
DatePicker 	This extends the standard TextBox control where date input is required. The DatePicker is a jQuery UI control. It provides a popup calendar control for easy date selection in date formats suitable for EAE and AB Suite.
Calculator 	This extends the standard TextBox control where a calculator is required. The Calculator control provides a dropdown calculator for easy calculation of a value to be inserted into the field. Note: Before using the Calculator control, the calculator.css and calculator.js files must be enabled in the MainUIPage.html file.

Hyperlink Client Tools Consultancy	This extends the standard Label control. The Hyperlink custom control is used for substituting label controls where the label text has been specified as the html anchor control. I.e.: `<A HREF ='http://www.ClientTools.com.au ' >Client Tools Consultancy</A>`
Kendo UI Chart Control 	This extends the standard ListBox control. It allows the use of the Chart control provided by the third party Kendo UI. The Kendo UI Chart control provides easy charting capabilities of various chart types such as Column, Bar, Line and Pie. See the ReadMe file for instructions on how to obtain and install Kendo UI DataViz Controls.
Kendo UI ComboBox 	This extends the standard ComboBox control. It allows the use of the ComboBox control provided by the third party Kendo UI as an alternative to the standard ComboBox control. The Kendo UI ComboBox provides a different look and feel. It supports Dropdown Styles 'dropdown' and 'dropdownlist' only. See the ReadMe file for instructions on how to obtain and install Kendo UI Web Controls.
Map Popup 	This extends the standard TextBox control. The Map Popup control can be attached to a group of address fields and show the address on a map. The Map Popup control is utilizing the Google JavaScript Map control. For conditions on using the Google Map control, see: https://developers.google.com/maps/documentation/javascript/tutorial
Slider	This extends the standard TextBox control. The Slider is a jQuery UI control. The Slider changes a TextBox control to a graphical slider that allows the user to choose a numeric value from a range. The Slider's orientation can be horizontal or vertical.

	
CopyFrom List 	This extends the standard CopyFrom Grid control. The CopyFrom List control can substitute a CopyFrom area in which all fields are inquiry only fields. The CopyFrom List control will create a list of all rows in the CopyFrom area by automatically returning to the host as many times as required to retrieve all rows for the CopyFrom. The list will be displayed as a DataGrid allowing the user to scroll through the data and sort on columns. The CopyFrom List control requires the 'CopyFrom As Grid' option to be set to true.
	This extends the standard Check Box and Radio Button control. By setting the option 'CheckBoxRadioButton CustomControl' to true, the standard icons of the Check Box and Radio Button will be changed to custom icons that can be styled to suit individual styles. The icons scale up/down with the font size and grid lines can be applied.
Maint Buttons 	This extends the standard Button Group control. This control allows generating customized buttons for the Maint field including icons on buttons and custom style. With this control the Maint buttons can be positioned anywhere on the User Interface. This allows positioning the buttons in a fixed position outside of the form enabling quick access to the Maint buttons regardless of whether the ispec form is scrolled.
Input Mask 	This extends the standard TextBox control. The Input Mask control shows a mask in the text input box of a predefined format, making it easy for the user to enter data in the correct format. The control is based on the jquery "RobinHerbots/jquery.inputmask" plugin. For details on how to use the control see: https://github.com/RobinHerbots/jquery.inputmask Before using the Input Mask control, the jquery.inputmask.js files must be added to the MainUIPage.html file.
Field Masking	This extends the standard TextBox control.

Examples: Data: 123456789 Masked value: *****6789 Data: 123-45-6789 Masked value: XXX-XX-6789 Data: 123456789 Masked value: XXXXX6789	Field Masking is used for masking values entered and displayed in a TextBox control. This allows obscuring values such as Social Security Numbers and Credit Card Numbers. The value is shown unmasked when placing focus on the control allowing to modify the text. The value is shown masked when leaving the control and focus is elsewhere. Generic Data Masking syntax/format: ctcMask:[FIELDNAME], maskOptions:{regex:'\\d(?=....)', flags:'g', mask:'*'} Add the above generic Data Masking format to the data-bind property of the TextBox Control Specifications and specify the regex expression and masking characters. Examples: ctcMask:[FIELDNAME], maskOptions:{regex:' ^\\d{5}', mask:' *****'} Replace first 5 digits with *. ctcMask:[FIELDNAME], maskOptions:{regex:' (?:\\d{3})-(?:\\d{2})-(\\d{4})', mask:' XXX-XX-$1'} Replace digits in the first 2 groups with X. ctcMask:[FIELDNAME], maskOptions:{regex:'\\d(?=....)', flags:'g', mask:'*'} Replace all characters with * except last 4 characters.
Kendo UI Gauges Radial Gauge Linear Gauge 	This is an example of additional controls that can be added to the form at generate time. This utilizes the Generic User Control specifying the template for the gauge control.
TextList	This extends the standard ListBox control. A TextList control converts the rows of a listbox to text and displays the text in a read-only html DIV tag.

8.2 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 standard HTML 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 Smart Client Configurator** documentation.

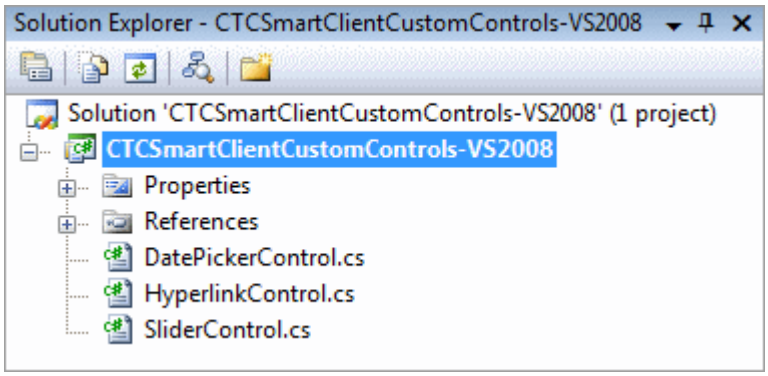
When creating Custom Controls, there are two distinct areas that need to be considered:

- The Generate Side
- The Runtime Side

8.2.1 Custom Controls – The Generate Side

Included with the installation of the CTC Smart Client Generator is a sample Visual Studio project that provides 4 examples of how to implement the Generate Side of Custom Controls.

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



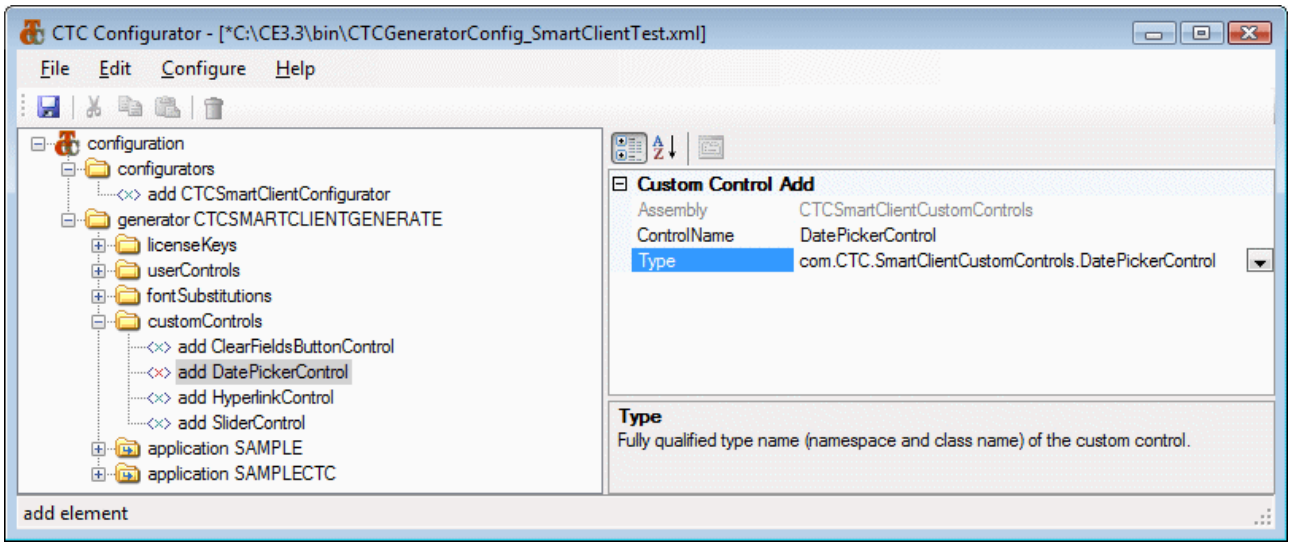
When building the CTCSmartClientCustomControls.csproj, a CTCSmartClientCustomControls.dll is created and added to the bin directory of the [ceroot] folder. It may be necessary to modify the CTCSmartClientCustomControls.csproj to point to the [ceroot] directory on the local machine.

Before customizing the Custom Controls copy the whole project to another location to avoid the changes being overwritten when next installing a new release of the CTC Smart Client Generator.

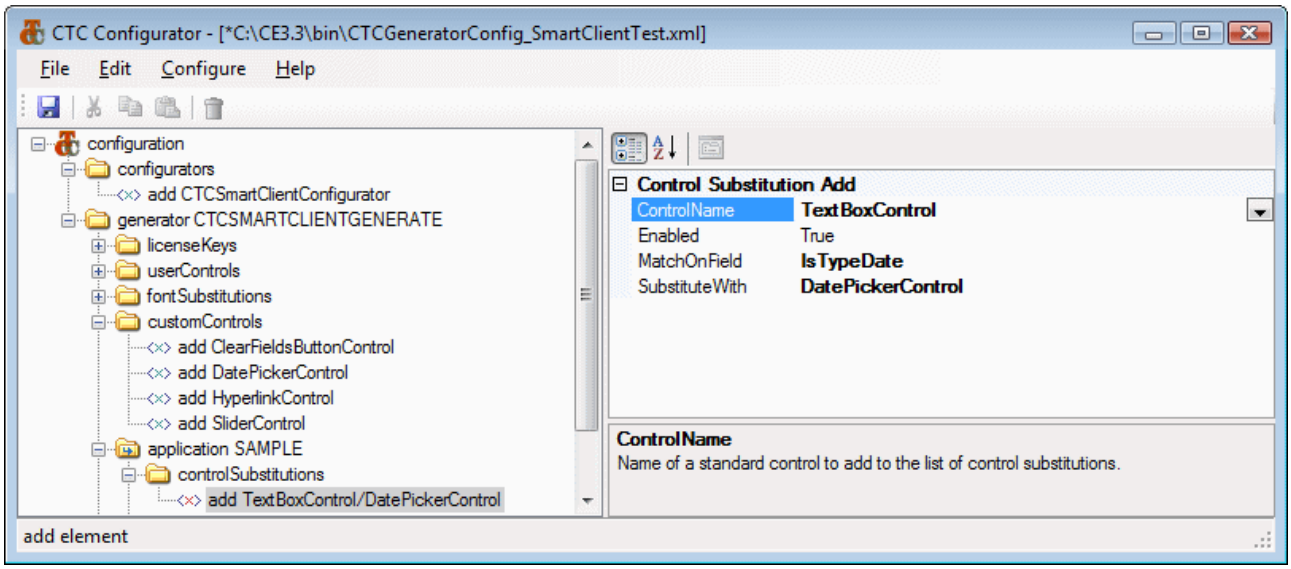
The DatePicker control is an example of how to implement a custom control. The DatePicker class implements the Generate Side and extends the CTC Standard TextBox Control to re-use as much of the generation of the TextBox control as possible. For the generation of a control, one method needs to be implemented:

- RenderControlSpecifications()
  This method outputs the specifications of the control to the .html file. The specification defines the look and feel and the data binding.

The following is an example of the DatePicker control 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.



8.2.2 Custom Controls - The Runtime Side

If specific runtime behaviour of a Custom Control, such as handling events or data converters, is required, it can be implemented on either the IspecViewExtended.js module or the ViewModelExtended.js class. Event handlers of a custom control would be implemented on the IspecViewExtended.js class as the Views/Forms inherit from this class. Data converters would be implemented on the ViewModelExtended.js class as the View Models inherit from this class.

