

CTC WPF Client Generator

Version 1.0.0



Table of Contents

1	Introduction	3
1.1	What is CTC WPF Client Generator?	3
1.2	The Concept	3
1.3	Standard Controls.....	4
2	Generator Initial Setup	5
2.1	EAE Setup.....	5
2.2	AB Suite Setup	6
2.3	Installing Bundle Infrastructure Files	8
3	Configuring the Generator	9
3.1	Generator Options.....	9
3.2	Generating CopyFrom As Table.....	13
3.3	Runtime Configuration	15
3.4	Control Specifications.....	15
4	The Generated User Interface Application	15
4.1	The Visual Studio Solution.....	15
4.2	The Generated Ispec Forms/Views.....	16
4.3	The Generated View Models	17
5	CTC WPF View Controller.....	18
6	Deployment Options.....	20
6.1	Executable Application	20
6.2	Downloadable Files.....	21
6.3	Configuration File Location	22
7	Custom Controls	22
7.1	System Provided Custom Controls.....	22
7.2	Creating Own Custom Controls	24
7.2.1	Custom Controls – The Generate Side.....	24
7.2.2	Custom Controls - The Runtime Side	25

1 Introduction

1.1 What is CTC WPF Client Generator?

The CTC WPF Client Generator is a tool for creating a WPF user interface for EAE 3.3 and AB Suite systems.

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 WPF 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 WPF 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.

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 WPF 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 WPF 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 WPF 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 WPF 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 WPF 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	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.
Grid	DataGrid container for CopyFrom regions.
GridHeaderRow	DataGrid Header Row container for CopyFrom regions.
GridHeaderCell	DataGrid Header Cell container for CopyFrom regions.
GridRow	DataGrid Row container for CopyFrom regions.
GridCell	DataGrid 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	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.
Rectangle	Rectangular box.
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.

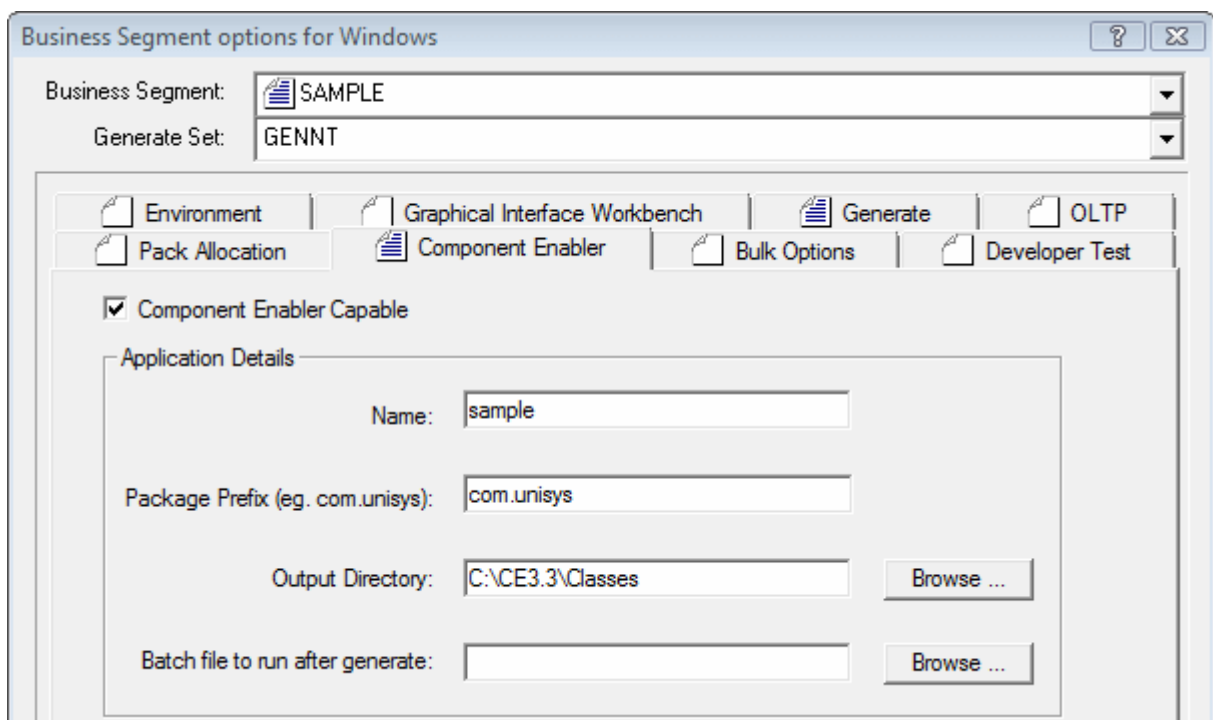
TeachText	Text for the teach form.
TextBox	Text input and Password box.

2 Generator Initial Setup

The CTC WPF 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 WPF 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.

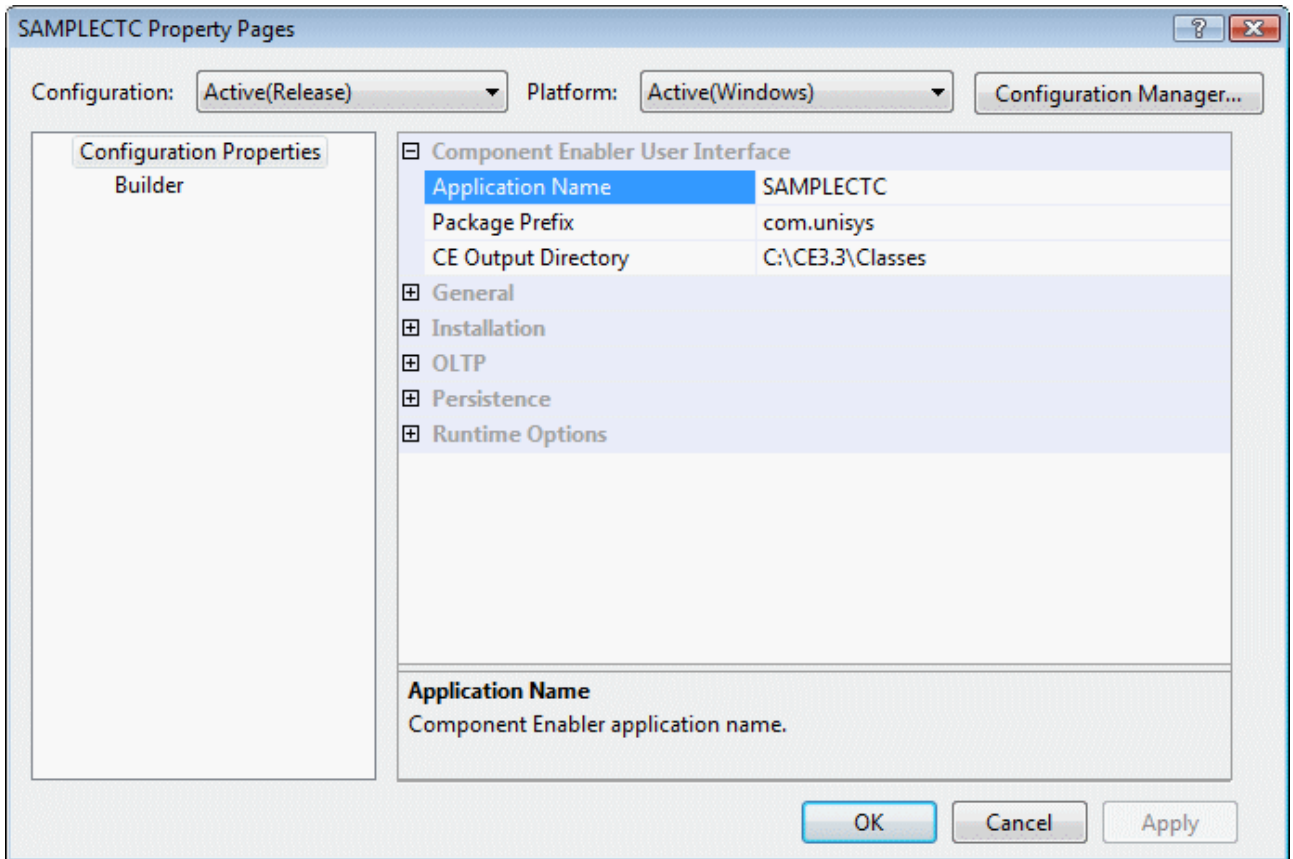
In addition to installing the CTC WPF Client Generator, users of EAE must also install the CTC Generate Gateway. The CTC Generate Gateway provides the necessary interface allowing the EAE Generate Environment to invoke generators implemented in .NET.

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. For further information see the ReadMe file of 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.

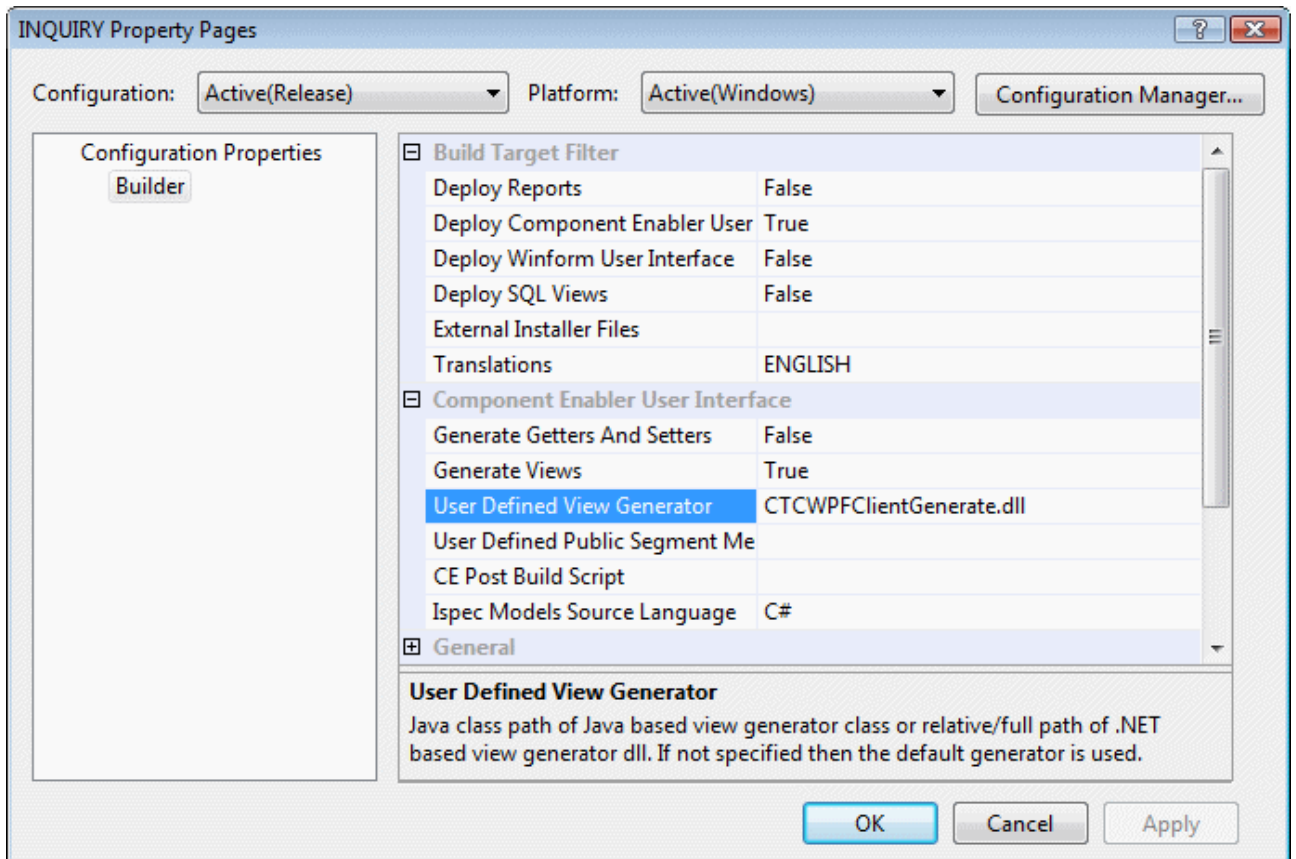
2.2 AB Suite Setup

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 WPF 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.

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 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 files 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 'CTCWPFClientInfrastructureFiles.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.

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 WPF 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 WPF while Developers concentrate on the back-end implementation.

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.

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 CTC Standard 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 WPF.

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 Type

This option specifies the type of CopyFrom table to generate.

The 'Table' option generates the CopyFrom region as a traditional table based on the WPF Grid Layout control.

The 'Grid' option generates the CopyFrom region as a WPF DataGrid control. The DataGrid control provides properties that make it easy to customize the look and feel of the CopyFrom table.

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.

- `AfterInitializeForm ()`: This method is invoked as part of initializing the form after the standard `InitializeForm()` method is invoked. This method allows adding any additional form initializing code as required.
- `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.

- `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 an extension.

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.

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.

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 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.

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.

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 Validate Numeric

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

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 hovers over the control.

Visual Studio Version

This option specifies the Visual Studio version. This is currently a fixed value 'VS2008' which cannot be changed.

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 WPF Client 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:

The diagram shows a rendered table with the following data:

Most Recent Transactions:						
Ref	Date	Time	Tran	Vend/Cust	Quantity	
2	23SEP09		1557	CNOTE	C1	1+
1	23SEP09		1556	CNOTE	C1	1+
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	

Callouts in the diagram point to the following controls:

- GridRow control (points to a data row)
- GridHeaderRow control (points to the header row)
- Grid control (points to the entire table)
- GridHeaderCell control (points to a header cell)
- GridCell control (points to a data cell)

The table above is an example of generating the CopyFrom region with the option CopyFromType = "Grid". The table is generated as a WPF DataGrid control. Each of the elements in the table (Grid, GridHeaderRow, GridHeaderCell, GridRow and GridCell) are standard CTC controls and can be styled using the CTC Configurator to suit local requirement.

Ref	Date	Time	Tran	Vend/Cust	Quantity
2	23SEP09	1557	CNOTE	C1	1+
1	23SEP09	1556	CNOTE	C1	1+
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 table above is an example of generating the CopyFrom region with the option CopyFromType = 'Table'. The table is generated as a traditional WPF Grid Layout control. Each of the elements in the table (Table, TableHeaderRow, TableHeaderCell, TableRow and TableCell) are standard CTC 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 WPF ScrollViewer control in order to limit the size and to provide scroll bars. This can be achieved using the CTC Configurator and modifying the specifications of the Table control to include a ScrollViewer 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. 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 as: "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 label controls, are not identified as CopyFrom controls by the generate environment and are therefore not generated as part of the table.

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 WPF Client Configurator** documentation for further details.

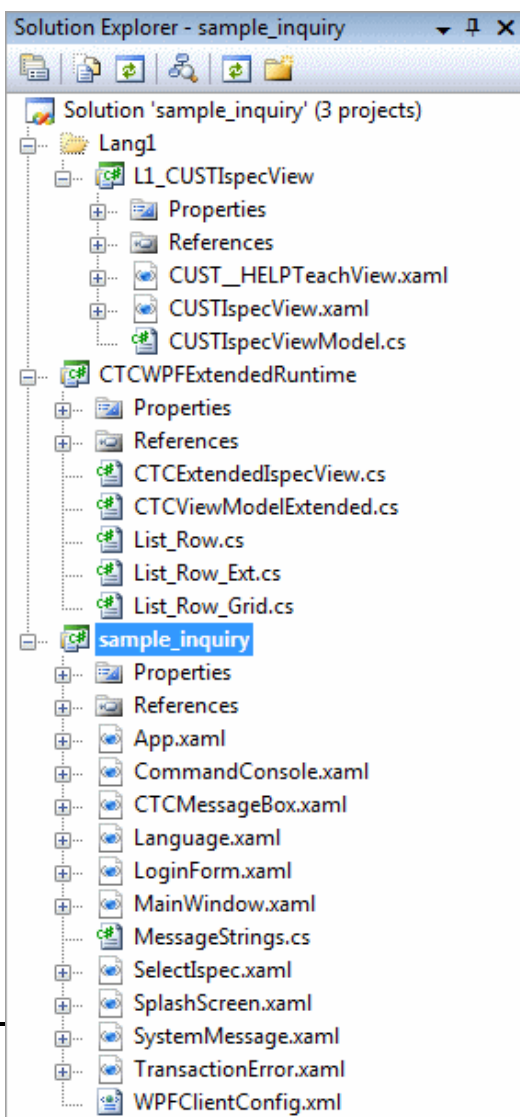
3.4 Control Specifications

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 WPF controls can be specified on the controls.

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

4 The Generated User Interface Application

4.1 The Visual Studio Solution



The generated application is a WPF application created for Visual Studio 2008. A WPF solution containing two fixed projects plus one generated project for each ispec in the bundle is created. The solution can be opened and inspected using Visual Studio 2008. To the left is an example of a generated solution with one ispec 'CUST' in the bundle.

The project named **sample_inquiry** represents the WPF client application. It contains fixed non-generated files required for the WPF application on the client side. The MainWindow.xaml file is the main window that opens when the user starts the application. The Main Window hosts the CTC WPF 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 WPF View Controller, see section below.

The WPFClientConfig.xml file contains the parameters to configure host system connectivity required by Component Enabler.

The project named **CTCWPFExtendedRuntime** contains files that allow the customization of the

runtime behaviour of the application by extending the default behaviour provided by CTC.

All files in the two projects 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 project named **L1_CUSTIspecView** contains generated files for the CUST ispec. One project is generated for each ispec, which causes each ispec to be compiled into a separate dll. This allows small packages to be downloaded to the client, as well as the downloading of individual ispec forms only when they are required.

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

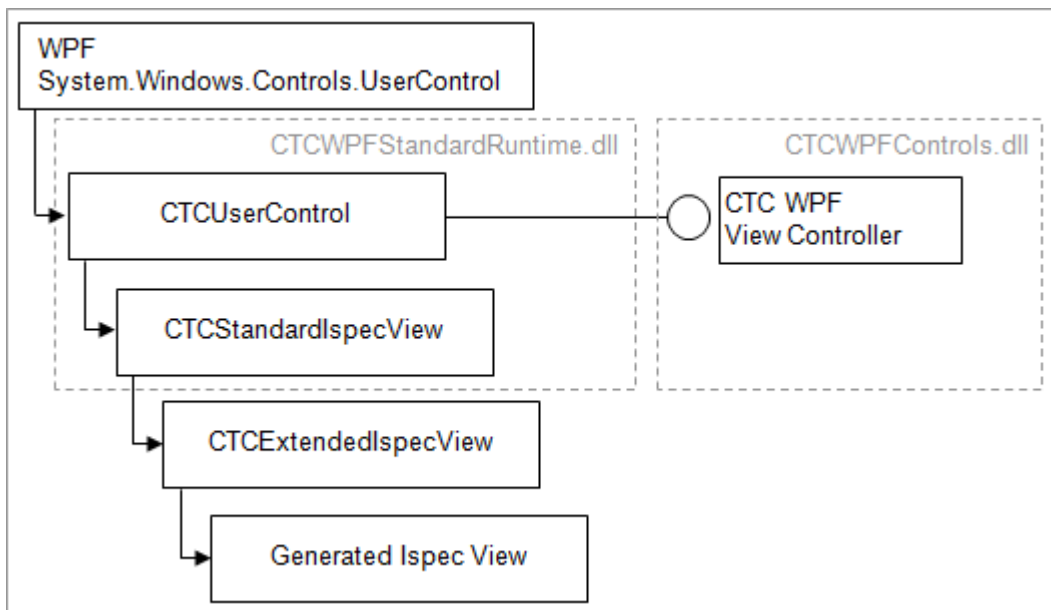
- TeachView - This is the teach/help information generated as a WPF XAML form.
- IspecView – This is the ispec form containing all controls as they are painted at design time in EAE or AB Suite Developer generated as a WPF XAML form. Controls on the form bind to properties on the IspecViewModel.
- IspecViewModel – This is a .NET class providing the interface between the controls on the form and the data held in the ispec model. This class contains one property for each data field defined on the ispec.

Ispecs are generated taking full advantage of the WPF data binding capabilities. Each control on the form uses the WPF 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 The Generated Ispec Forms/Views

The form/view generated for an ispec is created as a WPF User 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 CTCExtendedIspecView 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 CTCWPFEExtendedRuntime, 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.

The CTCUserController class provides the interface to the CTC WPF View Controller. It inherits from the Microsoft WPF System.Windows.Controls.UserControl class which provides the WPF User Interface behaviour.

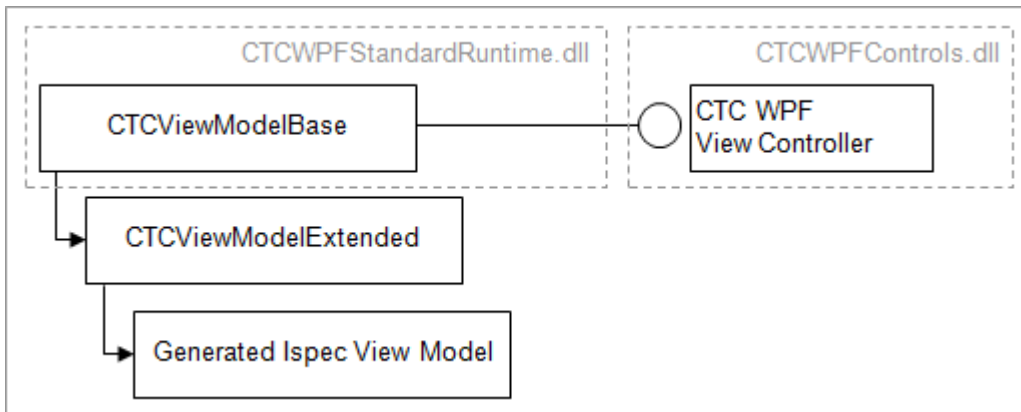
The CTCUserController and CTCStandardIspecView classes are supplied as a dll named CTCWPFEStandardRuntime.dll which is installed in the bin directory of the generate solution.

4.3 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 with a get and a set method is generated for each field defined on the ispec. The get/set method reads/writes the corresponding field value on the actual Component Enabler Ispec Model which provides the interface to 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 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 project named CTCWPFEExtendedRuntime, 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 WPF View Controller which provides access to the Ispec Model.

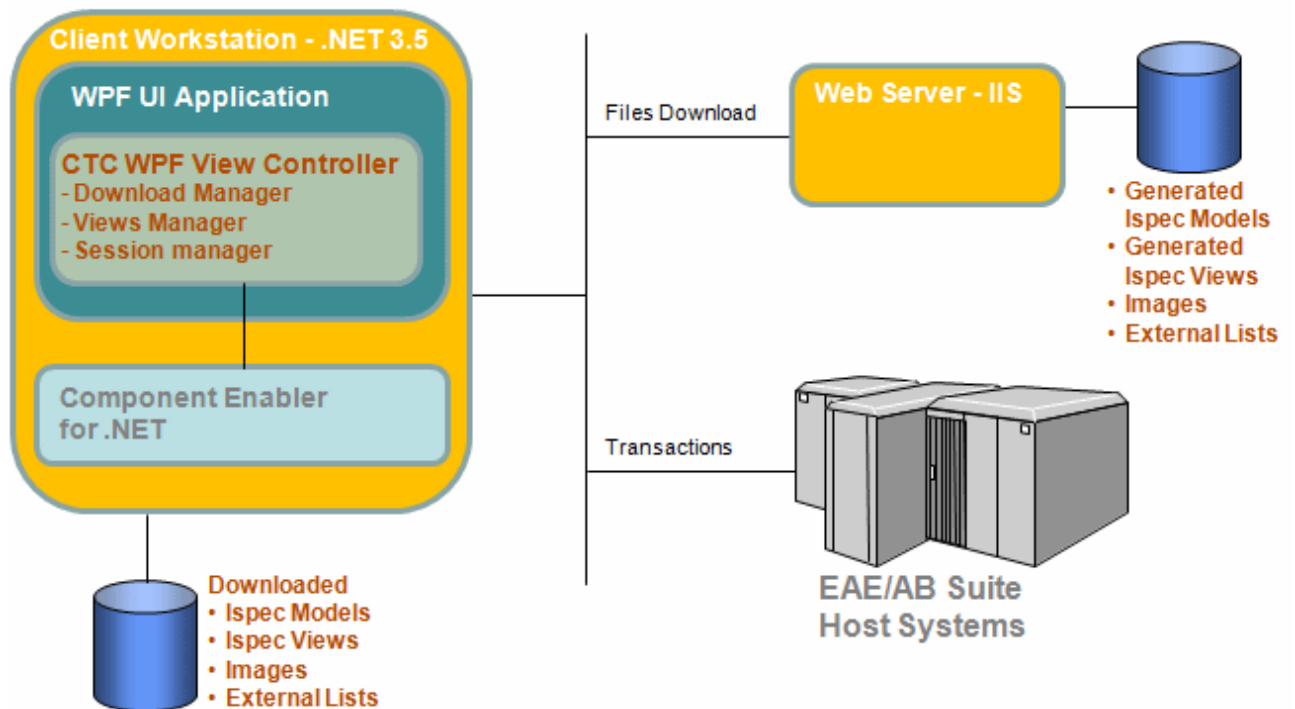
The CTCUserControl class provides the interface to the CTC WPF View Controller. It inherits from the Microsoft WPF System.Windows.Controls.UserControl class which provides the WPF User Interface behaviour.

The CTCViewModelBase class is part of the CTCWPFStandardRuntime.dll which is installed in the bin directory of the generate solution.

5 CTC WPF View Controller

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

Runtime Architecture of Generated WPF Application



The diagram above provides an overview of the runtime architecture of a WPF application generated by the CTC WPF 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 WPF 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.

The View Controller uses the standard Component Enabler from Unisys to communicate with the EAE/AB Suite host system.

The View Controller is configured using the WPFClientConfig.xml file. Configuration parameters can be set by editing the WPFClientConfig.xml file directly 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 WPF Client Configurator** document.

The CTC WPF View Controller includes a highly optimized Views Manager. Forms/views are downloaded dynamically from the web server as and when they are required, determined by user navigation through the application. Other items required by the application such as images and external lists are also downloaded dynamically. All items downloaded are cached locally on the client workstation. This enables downloading of items only when a new version is available on the web server. A check for a newer version of an item is performed once per session.

The WPF Main Window hosting the View Controller can control the behaviour of the View Controller by using the programmatic interface. The WPF Main Window can get control at key events during the end user interaction with the forms/views. The View Controller raises the following events:

- ViewLoad – Occurs when a form/view has been loaded and added to the visual tree and before it is displayed to the user.
- 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 to be sent to the host system. Setting the CancelViewRendering property on the event arguments 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 arguments 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.
- ViewLoad – Occurs when a view/form has been loaded into the visual tree and before the form is displayed to the end user. This provides the opportunity for the application to update, for example, the title of the window to show the name of the current form, or to adjust the background color of the form.
- AlternateView – Occurs when the end user navigates to a new form/view, just before loading the new form. This provides the opportunity for 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 binding to the data properties of the generated View Models as mentioned above, and still take full advantage of the capabilities of the CTC WPF View Controller and the CTC solution. Setting properties such as ViewName and ViewType on the event arguments specifies the name and type of the alternate view to load.

- **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.

6 Deployment Options

From a deployment point of view, the generated user interface application consists of two elements:

- **Executable Application.** This is the UI Application containing the Main Window which includes the CTC WPF View Controller that controls the loading of the ispec forms/views and communicates with the EAE/AB Suite host application. This application is located in the bin directory of the views directory of the generated bundle (i.e.: \views\sample-inquiry\bin\release\sample-inquiry.exe).
- **Downloadable Files.** These are the generated Ispec Model files, Ispec Forms/Views, Images and External Lists.

6.1 Executable Application

Before the end user can run the application the first time, the Executable Application and associated dll's must be installed on the end user workstation. The files required to be installed are located in the bin directory of the views directory of the bundle (i.e.: \views\sample-inquiry\bin\release):

- All *.dll
- [ApplicationName]_[BundleName].exe
- WPFClientConfig.xml
- As an alternative to installing the Component Enabler software on the end user workstation, the following two dll's can be included with the Executable Application:
 - [ceroot]\bin\CEdotNET.dll
 - [ceroot]\bin\CEWindowsAPI.dll

These files can be installed in a number of ways:

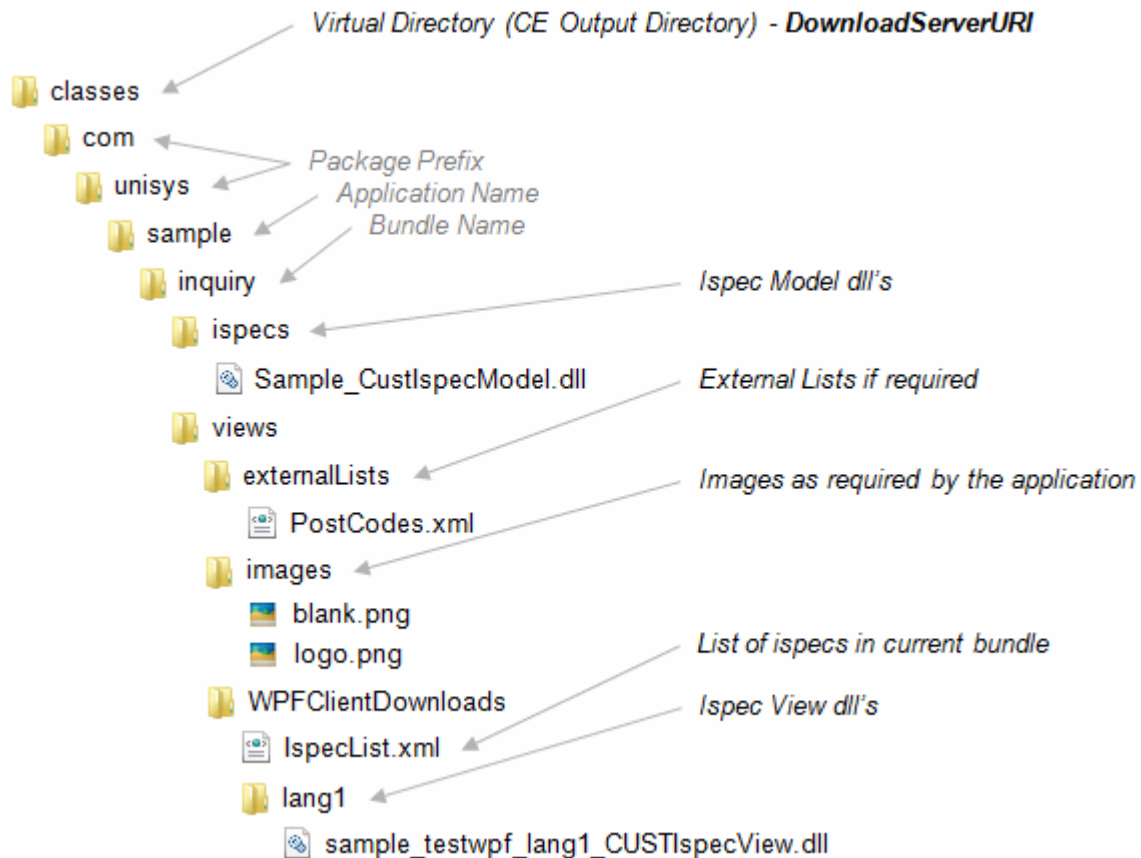
- **Manual Deployment.** Copy the required files to a directory on the end user workstation.
- **Deploy via a traditional setup program.** This option involves creating a Microsoft Installer (MSI) setup. Creating a MSI setup allows you to provide a fully fledged automated installation including the creation of a shortcut to the application. To assist in creating a MSI setup, Visual Studio includes a Setup Project that can be added to the generated solution.
- **ClickOnce Deployment.** WPF applications integrate with the ClickOnce setup feature of Visual Studio, which allows users to launch the setup program from the browser. Applications that are installed through ClickOnce can be configured to automatically check for updates. ClickOnce can be configured using Visual Studio by starting the Publish Wizard from the generated application project.

For prerequisites of the end user workstation, see the **CTC WPF Client Generator ReadMe** document.

6.2 Downloadable Files

Distribution of files, such as the generated Ispec Models and Ispec Views that change with every generate of the application, and other files such as Images and External Lists that are external to the application, can be automated by deploying these files to a virtual directory on a web server.

The directory structure on the web server must match the directory structure of the CE Output Directory. The diagram below shows the directory structure and outlines the files to be copied/deployed to the web server:



The following outlines the major steps involved in preparing the web server:

- A virtual directory on the web server pointing to the CE Output Directory as illustrated above must be created.
- The configuration parameter **DownloadServerURI** must be set to the name of the virtual directory.
- A directory structure matching the CE Output Directory as illustrated above must be created.
- The following files must be copied from the generate workstation to the web server as illustrated above:
 - All generated Ispec Model dll's
 - Any external lists as required by the application
 - All images as required by the application

- The IspecList.xml file
- All generated Ispec View dll's

While the above distribution method is the recommended method, distributing these files via a web server is optional. The directory structure including the files as listed above can be manually copied to the end user workstation or included with the installation of the main executable application and nothing will be downloaded.

6.3 Configuration File Location

By default, the Executable application (MainWindow.xaml) delivered with the Generator reads the configuration parameters from the WPFClientConfig.xml file located in the same directory as the Executable application.

However, the location of the configuration file can be specified using the **ConfigFile** parameter on the application shortcut. This allows the configuration parameters to be maintained in a central location without having to redistribute the configuration file whenever it changes.

For example, the following command line

Sample_Inquiry.exe /ConfigFile:http://WebServerName/VirDir/WPFClientConfig.xml on the shortcut specifies to read the configuration parameters from the WPFClientConfig.xml file located in the virtual directory VirDir on the web server WebServerName.

7 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 Control (see the **CTC WPF Client Configurator** documentation for further details).

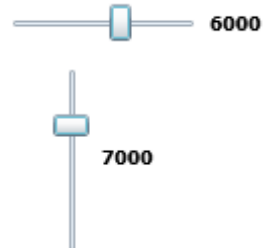
Included with the CTC WPF Client Generator are a number of Custom Controls. These can be used out of the box or changed to suit local requirement.

7.1 System Provided Custom Controls

Below is a list of Custom Controls that are delivered as part of the CTC WPF Client Generator.

Included in the list are a number of controls from the Microsoft WPF Toolkit. The WPF Toolkit is a free download. Currently, four controls from the toolkit, which are particularly suitable for inclusion on EAE and AB Suite forms to enhance the user experience, have been included as custom controls. Other controls can be included on request. For more information and to download the latest release of the WPF Toolkit see <http://wpf.codeplex.com/>.

Control	Description
Charting	This extends the standard ListBox control. The Charting is a WPF Toolkit control. It provides easy

 <p>The top chart is a bar chart titled 'Product Sales' showing 'Units Sold' for products P1 through P6. The y-axis ranges from 0 to 8. The bars show values of approximately 8 for P1, 2 for P2, 6 for P3, 5 for P4, 7 for P5, and 1 for P6. The bottom chart is a pie chart titled 'Product Sales' showing the distribution of units sold across products P1 through P6. The legend indicates colors for each product: P1 (blue), P2 (red), P3 (green), P4 (yellow), P5 (purple), and P6 (dark blue).</p>	<p>charting capabilities of various chart types such as Area, Bar, Bubble, Column, Line, Pie and Scatter.</p>																														
<p>DataGrid</p> <table border="1" data-bbox="236 792 630 992"> <thead> <tr> <th>Name</th> <th>Region</th> </tr> </thead> <tbody> <tr> <td>Sam Johnson</td> <td>Asia</td> </tr> <tr> <td>Peter Phillips</td> <td>Australia-NZ</td> </tr> <tr> <td>Robert Campbell</td> <td>Europe</td> </tr> <tr> <td>Jenny Thomas</td> <td>North America</td> </tr> <tr> <td>Sandra Knight</td> <td>South America</td> </tr> <tr> <td>Allan Parker</td> <td>Other</td> </tr> </tbody> </table> <table border="1" data-bbox="236 1003 630 1144"> <thead> <tr> <th>Image</th> <th>Product</th> <th>Product Description</th> <th>Units Sold</th> </tr> </thead> <tbody> <tr> <td></td> <td>P6</td> <td>Sony A-200 D-SLR/18-70mm Kit</td> <td>1</td> </tr> <tr> <td></td> <td>P2</td> <td>Panasonic Lumix FZ8</td> <td>2</td> </tr> <tr> <td></td> <td>P4</td> <td>Canon EOS450D/EF50-55IS/EF75</td> <td>6</td> </tr> </tbody> </table>	Name	Region	Sam Johnson	Asia	Peter Phillips	Australia-NZ	Robert Campbell	Europe	Jenny Thomas	North America	Sandra Knight	South America	Allan Parker	Other	Image	Product	Product Description	Units Sold		P6	Sony A-200 D-SLR/18-70mm Kit	1		P2	Panasonic Lumix FZ8	2		P4	Canon EOS450D/EF50-55IS/EF75	6	<p>This extends the standard ListBox control.</p> <p>The Data Grid control is a flexible control with many options such as column headers, column sorting, column resizing, column reordering and styling.</p>
Name	Region																														
Sam Johnson	Asia																														
Peter Phillips	Australia-NZ																														
Robert Campbell	Europe																														
Jenny Thomas	North America																														
Sandra Knight	South America																														
Allan Parker	Other																														
Image	Product	Product Description	Units Sold																												
	P6	Sony A-200 D-SLR/18-70mm Kit	1																												
	P2	Panasonic Lumix FZ8	2																												
	P4	Canon EOS450D/EF50-55IS/EF75	6																												
<p>DatePicker</p> <p>12/05/2009 15</p> 	<p>This extends the standard TextBox control where date input is required.</p> <p>The DatePicker is a WPF Toolkit control. It provides a popup calendar control for easy date selection in date formats suitable for EAE and AB Suite.</p>																														
<p>Slider</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 range. The Slider's orientation can be horizontal or vertical.</p>																														

7.2 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 WPF 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 WPF Client Configurator** documentation.

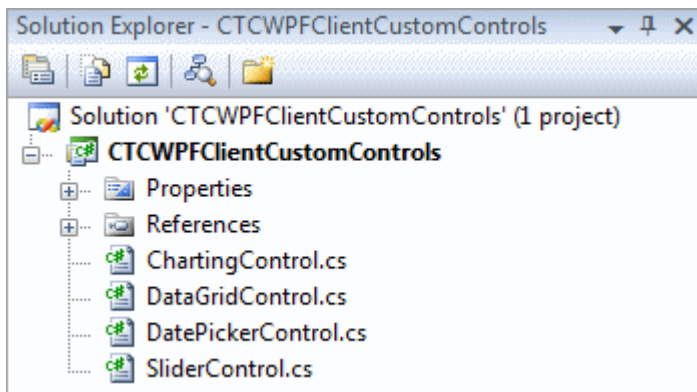
When creating Custom Controls, there are two distinct areas that need to be considered:

- The Generate Side
- The Runtime Side

7.2.1 Custom Controls – The Generate Side

Included with the installation of the CTC WPF 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 **CTCWPFClientCustomControls.csproj** and is installed into the **CustomControls** directory of the [ceroot]\CTC-Software folder. The project is shown below.

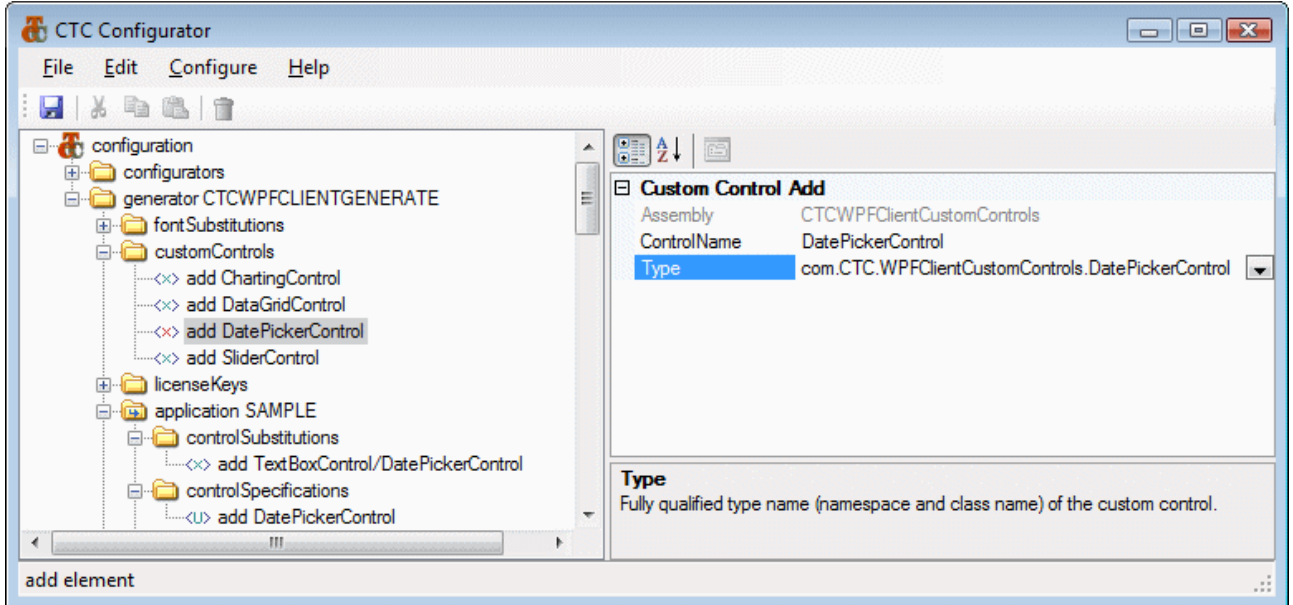


When building the CTCWPFClientCustomControls.csproj, a CTCWPFClientCustomControls.dll is created and added to the bin directory of the [ceroot] folder. It may be necessary to modify the CTCWPFClientCustomControls.csproj to point to the [ceroot] directory on the local machine.

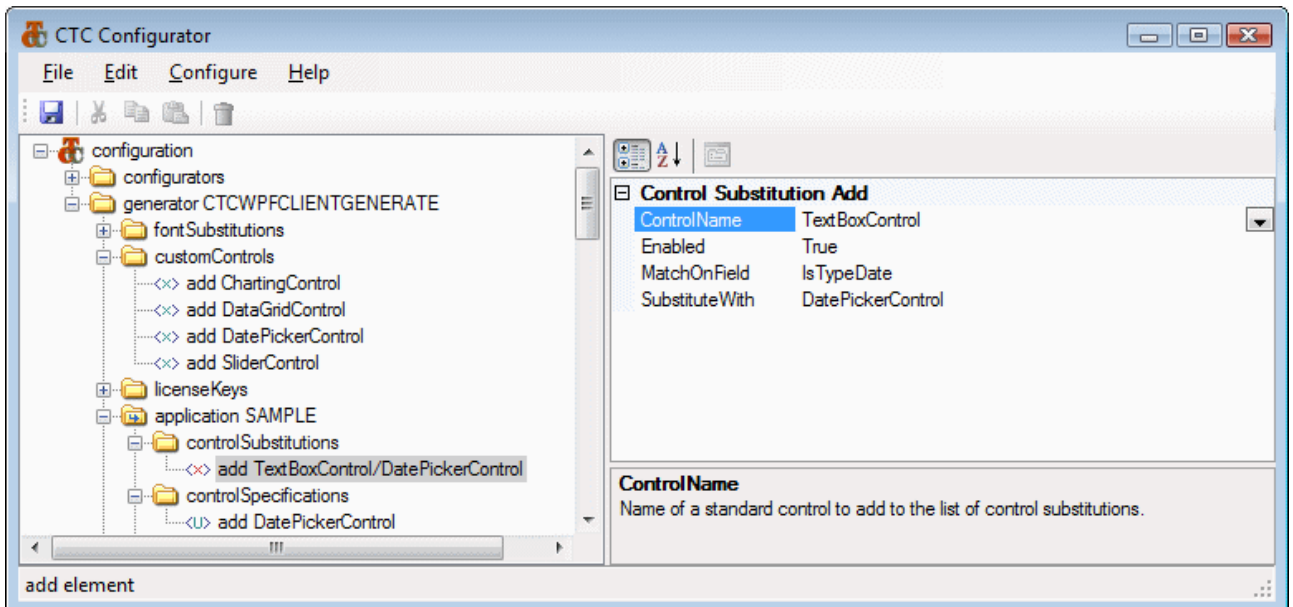
The DatePicker control is an example of how to implement a custom control. The DatePicker class implements the Generate Side and extends the CTC Standard TextBox Control to re-use as much of the generation of the TextBox control as possible. For the generation of a control, one method needs to be implemented:

- `RenderControlSpecifications()`
This method outputs the specifications of the control to the .xaml 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.



7.2.2 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 `CTCExtendedISpecView` class or the `CTCViewModelExtended` class. Event handlers of a custom control would be implemented on the `CTCExtendedISpecView` class as the Views/Forms inherit from this class. Data

converters would be implemented on the CTCViewModelExtended class as the View Models inherits from this class.