# CTC WPF Client Configurator

Version 2.0.1

## Table of Contents



# 1 Introduction

## 1.1 What is CTC WPF Client Configurator?

The CTC WPF Client Configurator is the Configurator for the WPF Client Generator from Client Tools Consultancy.

The Configurator is an add-in to the CTC Configurator Framework. It provides the user interface to configure options and features specifically for the WPF Client Generator.

This document should be read in conjunction with the **CTC Configurator Framework** document and the **CTC WPF Client Generator** document.

## 1.2 The Concept

Typically, a generator creates a fixed, pre-determined user interface application, where users have limited or no influence on what is generated. 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.

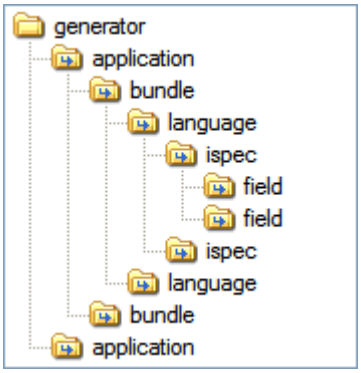
This requires the generator to be very flexible and to provide the means for customizing the result of the generate prior to entering the generate phase.

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.

## 1.3 Configurable Elements

A configuration consists of configurable elements, and options and features that can be specified for each of these elements. Configurable elements are generator, application, bundle, language, ispec and field, which are specified in a hierarchy as illustrated below.



For information about configurable elements and their hierarchical relationship, refer to the help documentation for the **CTC Configurator Framework**.

The following is a list of items (options/features) that can be configured for the WPF Client Generator within each of the configurable elements (generator, application, bundle, language, ispec and field):

- License Keys
- Custom Controls
- Control Substitutions
- Control Specifications
- Font Substitutions
- Options
- Runtime Configuration

## 1.4 License Keys

License Keys are required for using the generator and can be specified at the generator level only.

License keys for the WPF Client Generator are specific for the computer on which the generator is installed. A license key is obtained by contacting Client Tools Consultancy.

License keys are stored in this element. When starting the WPF Client Generator for the first time it will request the user to enter a valid license key, evaluation license or full license. Alternatively, the license key can be entered directly into the configuration by adding a License key element to the WPF Client Generator.

When sharing configuration files between work stations by copying a configuration file, the license key element can contain keys that are invalid on the current computer. Invalid license keys are ignored by the generator.

Invalid or expired license keys can be deleted.

## 1.5 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 or they can be customized.

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 Configuration Items

## 2.1 Custom Controls

Custom Controls are controls created by either the customer or CTC for a specific purpose. A Custom Control is used when none of the Standard Controls match the requirements.

Custom Controls also allow the utilization of third party controls in the generated user interface application. For example, to utilize a Date Picker control from a third party, a custom control can be created as an add-in to the generator, which would generate the correct source specifications and code for the third party control to be executed at runtime in the generated user interface application.

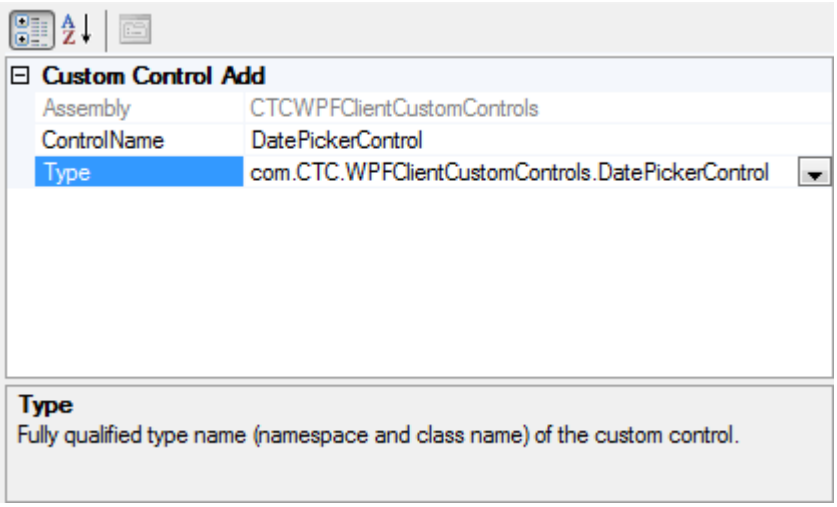
  

For information on how to create a Custom Control, refer to the **CTC WPF Client Generator** documentation.

A Custom Control is added to the configuration at the generator level only and applies to all configurable items such as application, bundle, ispec and field items within the generator.

The assembly dll containing the Custom Control must be copied to the bin sub-folder of the Component Enabler Root directory, for example, C:\CE3.3\bin.

When adding a Custom Control, the ControlName and Type properties as illustrated below must be specified.




**Assembly**: This identifies the assembly dll which contains the Custom Control. This is automatically filled when selecting the type.

**ControlDllLocation**: This specifies the path where the dll of the third party control is installed. This property is only required when the Custom Control is implementing a control that is not part of the standard set of WPF controls.

**ControlName**: User specified unique name of the Custom Control. When adding a new control type, this is pre-filled with the type name of the control.

**Type**: This is selected from the dropdown list, which is automatically filled with a list of Custom Controls found within the bin sub-folder of the Component Enabler Root directory. The type specifies the fully qualified type name, which is the namespace and class name of the Custom Control.

## 2.2 Control Substitutions, Add

Control Substitutions are used when replacing/substituting Standard Controls with Custom Controls.

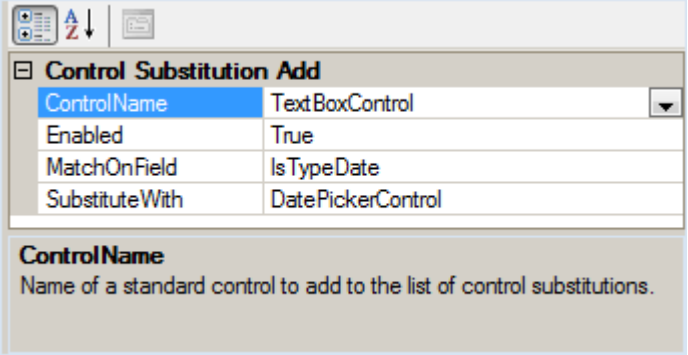
Control Substitutions can be added to any level in the hierarchy.

Controls being substituted must be of similar type. For example, when substituting a textbox control with a custom date popup control, the date popup control must be capable of understanding how to generate the control that was originally painted as a textbox.

When control substitutions are inherited from parent levels, control substitutions at the current level are evaluated first. When substituting the same control with different custom controls specifying match conditions, the substitutions will be evaluated in the order they are specified, except for unconditional substitutions which will be evaluated last.

When adding a Control Substitution, the ControlName and SubstituteWith properties as illustrated below must be specified.




**ControlName**: This specifies the name of a Standard Control to be substituted. This is selected from the dropdown list, which is automatically filled with a list of available Standard Controls.

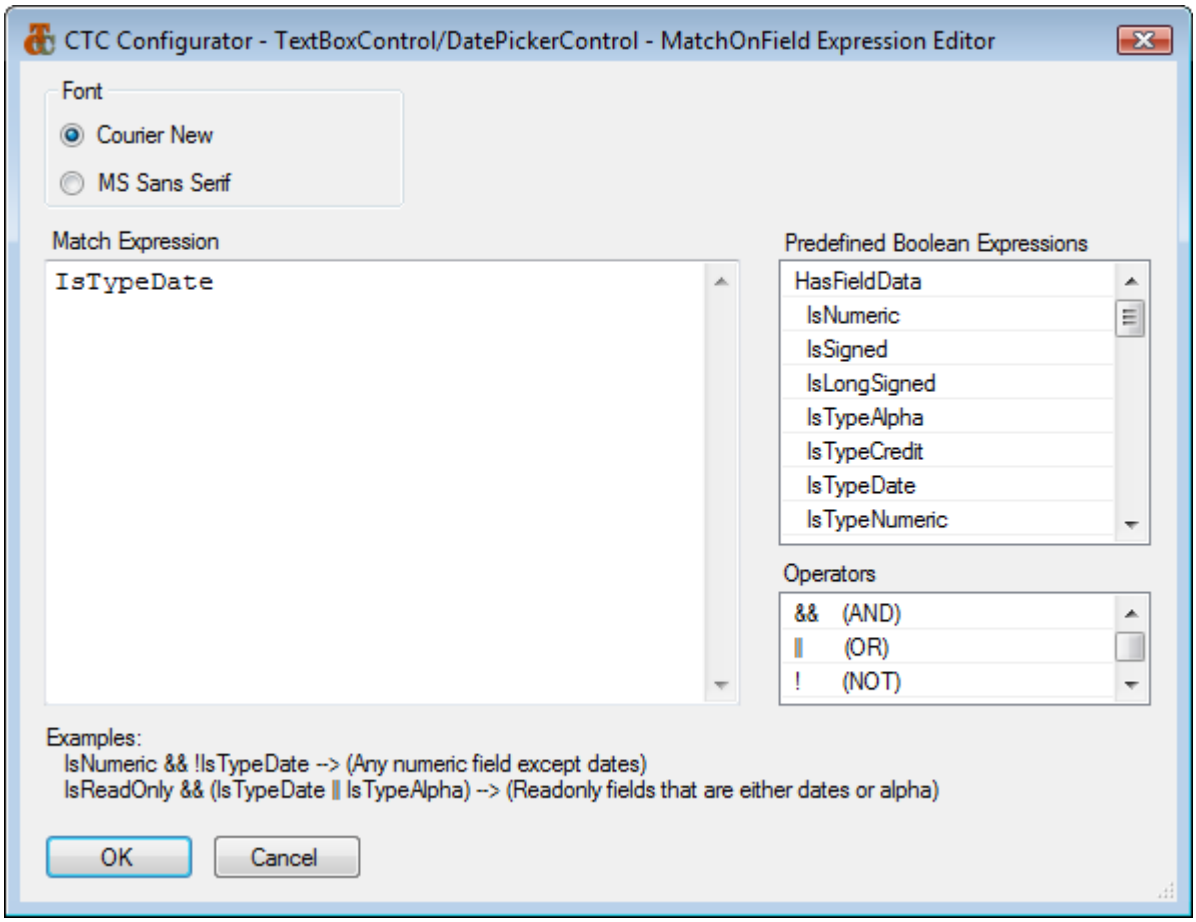
**Enabled**: This property provides a convenient way of removing a control substitution from the generate process without deleting it.

**SubstituteWith**: This specifies the name of a Custom Control to use instead of the Standard Control. This is selected from the dropdown list, which is automatically filled with a list of Custom Controls currently added to the generator.

**MatchOnField**: Optional match expression. MatchOnField is a boolean expression specifying the condition for selecting to which field or fields to apply the substitution. At generate time, the MatchOnField expression is evaluated against the field associated with the particular control and when the expression evaluates true, the substitution is applied. When no expression is specified, control substitution is unconditionally applied.

The expression is specified by selecting the ellipsis button . This will open the Expression Editor window as illustrated below.




An expression is specified using a selection of predefined boolean expressions. Expressions are dragged/dropped or copied/pasted from the list of predefined boolean expressions. Operators such as && (AND), || (OR), ! (NOT), == (Equal), != (Not Equal), > (Greater Than), < (Less Than), >= (Greater Than or Equal) or <= (Less Than or Equal) as well as parentheses ( ) can be used to create complex expressions.

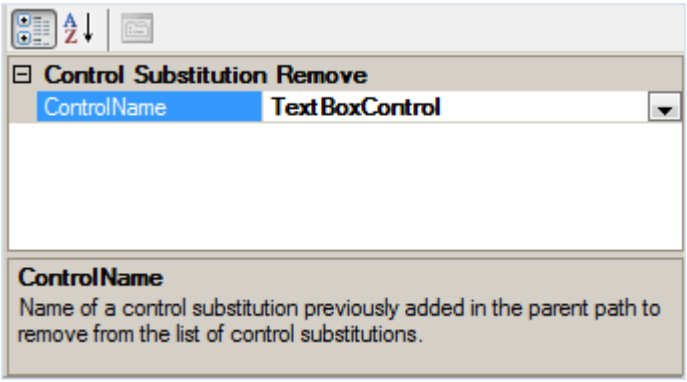
## 2.3 Control Substitutions, Remove

A Control Substitution that has previously been added to a parent in the hierarchy can be removed at any level in the hierarchy by inserting a Control Substitution Remove item.

For example, if a substitution of TextBoxControl with DatePopup has been added at the application level, it may be desirable to generate a specific field as a standard TextBoxControl and not as a DatePopup. In this case, a Control Substitution Remove item can be added to the field specifying to remove the substitution for the TextBoxControl.

The properties window of Control Substitution Remove is illustrated below.




**ControlName**: This specifies the name of a Standard Control that previously has been substituted at a parent level in the hierarchy. This is selected from the dropdown list, which is automatically filled with a list of available Standard Controls. For convenience, a 'Remove All' item is available in the list. When selecting 'Remove All', all control substitutions inherited from the parent levels will be removed.

## 2.4 User Controls

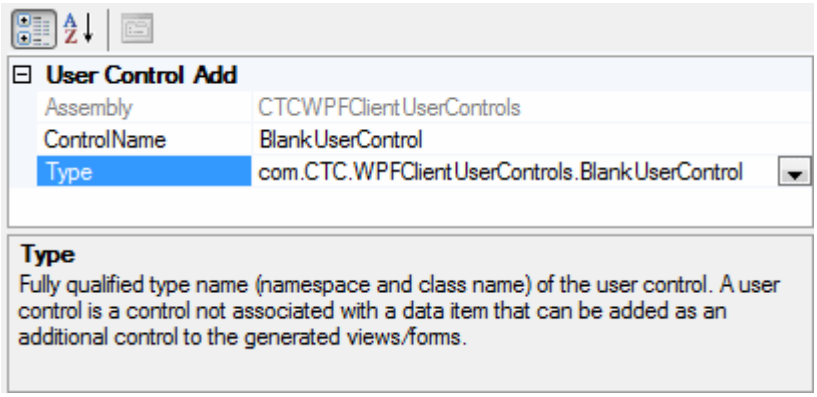
User Controls are controls created by either the customer or CTC for a specific purpose. A User Control is a control that is not associated with a data item in the EAE/AB Suite system. User Controls are used when additional controls are required on the forms to achieve specific tasks.

User Controls are similar to Custom Controls, except User Controls are not associated with data items in the EAE/AB Suite system. For information on how to create User Controls, refer to the 'Creating Own Custom Controls" section **CTC WPF Client Generator** documentation.

A User Control is added to the configuration at the generator level only and applies to all configurable items such as application, bundle, ispec and field items within the generator.

The assembly dll containing the User Control must be copied to the bin sub-folder of the Component Enabler Root directory, for example, C:\CE3.3\bin.

When adding a User Control, the ControlName and Type properties as illustrated below must be specified.




**Assembly**: This identifies the assembly dll which contains the User Control. This is automatically filled when selecting the type.

**ControlName**: User specified unique name of the User Control. When adding a new control type, this is pre-filled with the type name of the control.

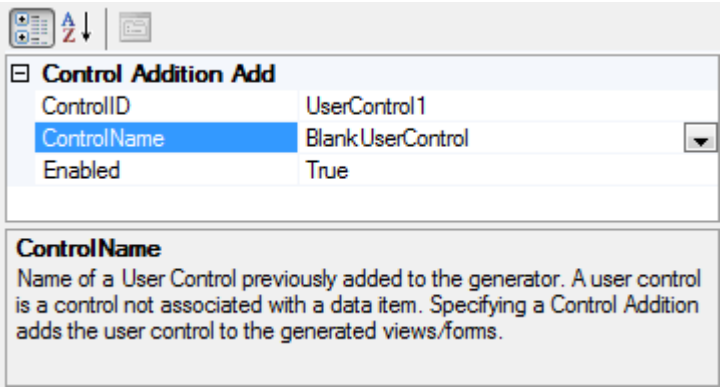
**Type**: This is selected from the dropdown list, which is automatically filled with a list of User Controls found within the bin sub-folder of the Component Enabler Root directory. The type specifies the fully qualified type name, which is the namespace and class name of the User Control.

## 2.5 Control Additions, Add

Control Additions are used when adding a User Control to a form.

Control Additions can be added to any level in the hierarchy.

When adding a Control Addition, the ControlName and ControlID properties as illustrated below must be specified.




**ControlID**: This specifies the ID of the User Control. This is automatically prefilled when a ControlName is selected. Control ID must be unique.

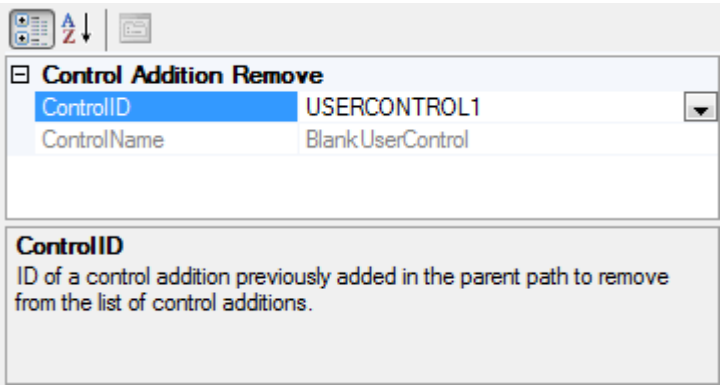
**ControlName**: This specifies the name of a User Control, previously added to the generator using the User Control Add option, to be added. This is selected from the dropdown list, which is automatically filled with a list of available User Controls.

**Enabled**: This property provides a convenient way of removing a control addition from the generate process without deleting it.

## 2.6 Control Additions, Remove

A Control Addition that has previously been added to a parent in the hierarchy can be removed at any level in the hierarchy by inserting a Control Addition Remove item.

The properties window of Control Addition Remove is illustrated below.



**ControlID**: This specifies the ID of a User Control that previously has been added at a parent level in the hierarchy. This is selected from the dropdown list, which is automatically filled with a list of available User Controls. For convenience, a 'Remove All' item is available in the list. When selecting 'Remove All', all control substitutions inherited from the parent levels will be removed.

**ControlName**: This specifies the name of the User Control associated with the ControlID. The ControlName property is read-only.

## 2.7 Control Specifications, Add

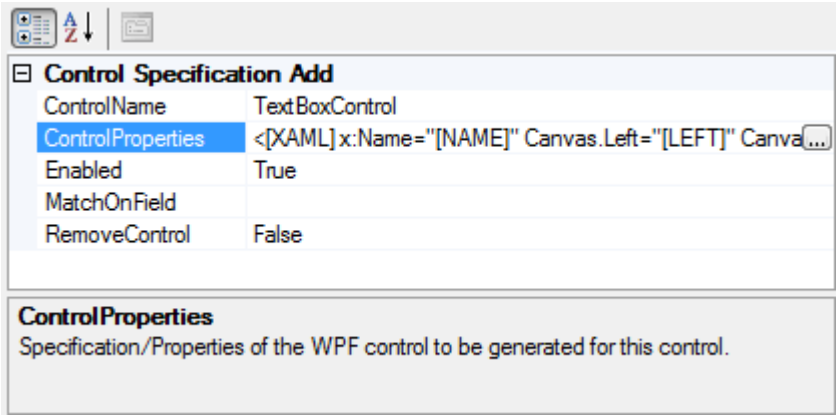
Control Specifications are used to change the default specifications of Standard Controls and Custom Controls. This allows modification of the default look and feel of a control by adding or removing properties and style attributes.

The specification of a control specifies the look and feel of the control. The properties of the control as it is generated are specified here.

Control Specifications can be added to any level in the hierarchy.

When control specifications are inherited from parent levels, control specifications at the current level are evaluated first. When specifying the same control with different match conditions, the specifications will be evaluated in the order they are specified, except for unconditional specifications which will be evaluated last.

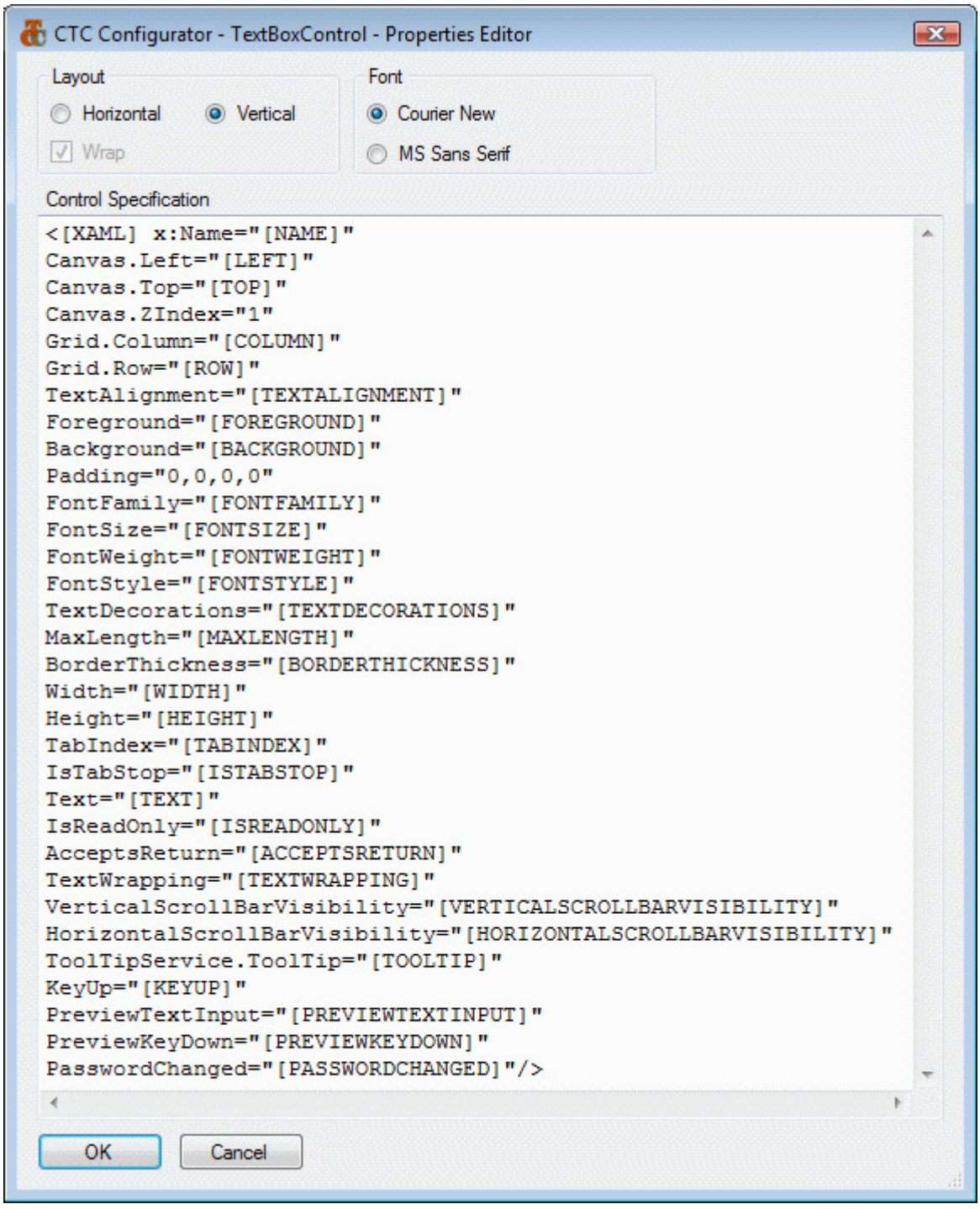
When adding a Control Specification, the ControlName and ControlProperties as illustrated below must be specified.




**ControlName**: This specifies the name of a Standard Control or a Custom Control for which to specify properties. This is selected from the dropdown list, which is automatically filled with a list of available Standard Controls and Custom Controls.

**ControlProperties**: This specifies the properties of the control. When selecting a control from the dropdown list of the ControlName property, the default specifications of the control will automatically be shown here. The properties are edited by selecting the ellipsis button ... which will open the Control Properties Editor window as illustrated below.




The specification of a control defines exactly how the control is generated. Via this editor, the user specifies the properties and their value to be set for the control.

The properties that can be specified on a control are those defined for the Microsoft WPF controls. Any property available on a WPF control can be added to the specifications of a CTC Standard Control.

The variables specified in square brackets represent the equivalent properties as they are specified for a control when painted in EAE or AB Suite. At generate time, the generator will replace the variables in square brackets with their actual value. If a variable has no value, i.e. it has not been specified in EAE or AB Suite, the whole property will be removed from the control at generate time.

The variables available for a control depend on the control. When initially selecting a control from the ControlName property, the default specifications of the control show the complete set of variables and their usage for the control as seen in the editor window above.

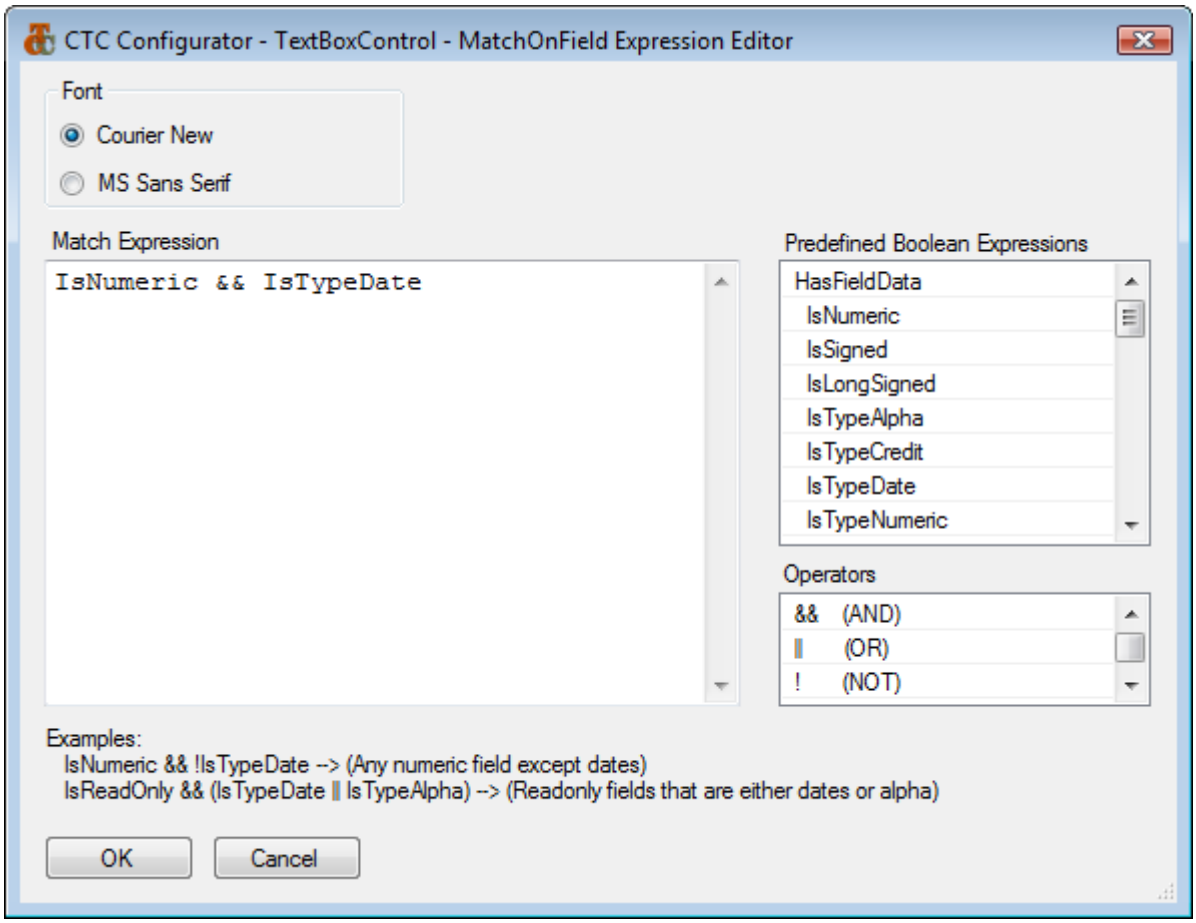
The variable names are case sensitive.

For easy readability, the editor window includes options for the user to choose layout and font.

**Enabled**: This property enables/disables the configuration of the specifications for a control. When false, the generator ignores this configuration entry. It provides a convenient way of removing a control specification from the generate process without deleting it.

**MatchOnField**: Optional match expression. MatchOnField is a boolean expression specifying the condition for selecting to which field or fields to apply the substitution. At generate time, the MatchOnField expression is evaluated against the field associated with the particular control and when the expression evaluates True, the substitution is applied. When no expression is specified, control substitution is unconditionally applied.

The expression is specified by selecting the ellipsis button [...] which will open the Expression Editor window as illustrated below.




An expression is specified using a selection of predefined boolean expressions. Expressions are dragged/dropped or copied/pasted from the list of Predefined Boolean Expressions. Operators such as && (AND), || (OR), ! (NOT), == (Equal), != (Not Equal), > (Greater Than), < (Less Than), >= (Greater Than or Equal) or <= (Less Than or Equal) as well as parentheses ( ) can be used to create complex expressions.

**RemoveControl**: Specifies whether this control is to be removed from the form. This is a convenient way of excluding a control from being generated.

### 2.7.1 Control Options

In addition to the WPF control properties, the following CTC Standard Controls have been extended with properties as options for the generation of behaviours specific to the CTC Standard Controls.

| Control | Property | Description |
|---|---|---|
| ComboBox<br>ListBox | List.XmlFilePath | List.XmlFilePath specifies an external XML formatted file to load and display in the list.<br>When specified, the file will automatically be retrieved from the server.<br>The path to the file is relative to the 'ClientBin' directory within the virtual directory where the application originates from. For example:<br>List.XmlFilePath="Countries.xml" points to http://localhost/Sample_Inquiry/ClientBin/Countries.xml |
| ComboBox<br>ListBox | List.XmlElementPath | List.XmlElementPath specifies the path to the xml element within the file holding the data to display in the list.<br>The path must be specified relative to the root element. For example:<br>List.XmlElementPath="country" specifies the country element within the root.<br>The XmlElementPath can specify a simple expression for the path to the element. For example:<br>List.XmlElementPath="country[@code='GB']/city" specifies the city element within the country which has the code attribute equal to GB.<br>When a complex expression is required, the expression must be specified in the code behind of the form that includes the list control. |
| ComboBox<br>ListBox | List.Key | List.Key is used for specifying the column in the list that is the key.<br>The key column holds the data that is sent to the host system.<br>For xml files, the Key property specifies an attribute name or element name within the XmlElementPath element. For example:<br>1) List.XmlElementPath="country" List.Key="code" or List.Key="@code" specifies the code attribute within the country element.<br>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:<br>List.Key=”1” specifies column 1 of the list as the key column. |
| ComboBox<br>ListBox | List.Columns | List.Columns is used for specifying the columns to include in the list and the data type of columns.<br>For xml files, the Columns property specifies the attribute names or element names within the XmlElementPath element. For example:<br>1) List.XmlElementPath=”country” List.Columns=”%code %name” or 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.<br>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.<br>For lists retrieved from the host system, the Columns property specifies the column numbers of the list. For example:<br>1) List.Columns=”%1 %2 %3” specifies to include columns 1, 2 and 3 of the list.<br>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.<br>The %-sign is used as the separator between column specifications.<br>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.<br>By default all columns are returned as string values. However, in special cases such as charting controls, it may be necessary to return numeric data as numeric values.<br>As an example, to return the third column in a list as a ‘Double’, the List.Column property is specified as follows:<br>List.Columns=”%1 %2 %3:Double”. |
| ComboBox<br>ListBox | List.MultiColumns | List.MultiColumns 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.<br>The default value is false and multi columns are concatenated into one column.<br>The MultiColumns property has only effect on lists retrieved from the host system. Xml files are always returned as multi columns. |
| ComboBox<br>ListBox | List.AddBlankRow | List.AddBlankRow 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.<br>By default, a blank row is added for ComboBox lists and no blank line is added for ListBox lists. |
| ComboBox<br>ListBox | List.Type | List.Type specifies the type of the list to return. By default, the generic 'com.CTC.SL.Runtime.List_Row' type is used. However, in special cases such as charting controls, it may be necessary to use a specialized type.<br>The 'com.CTC.SL.Runtime.List_Row_Ext', which can be found in the CTCWPFExtendedRuntime project, is an example of that. |
| ComboBox<br>ListBox | List.<br>DependentList | List.DependentList specifies the field name of a list whose content depends on the selection of this list.<br>When a list field name is specified, appropriate code will be generated to automatically clear and renew the list content of the dependent list.<br>The actual content displayed in the dependent list depends on the expression specified in the List.XmlElementPath property of the dependent list. |
| ComboBox<br>ListBox | List.XmlFilePath<br>List.Columns<br>List.Key<br>List.XmlElementPath | As an alternative to specifying the Xml List options parameters on the control specifications, the options can be specified in EAE/AB Suite Painter when painting a ComboBox or ListBox on a form.<br>The parameters can be specified as items for the control in the following order:<br>Item1: XmlFilePath<br>Item2: Columns<br>Item3: Key<br>Item4: XmlElementPath<br>The values must be specified within brackets. For example:<br>Item1: [Countries.xml]<br>Item2: [%code %name]<br>Item3: [code]<br>Item4: [country]<br>The items must be specified in the order as above.<br>Default values for items not specified can be specified |

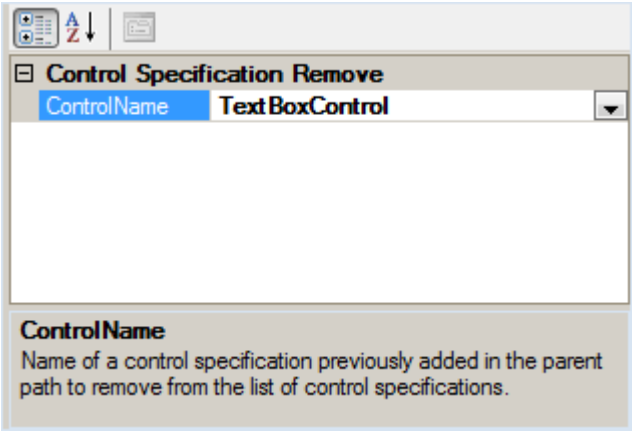
| | | |
|---|---|---|
| | | on the Control Specifications.<br>This requires the LableIdDetection option to be enabled. |
| TableRow | AlternatingBackColor | When specified, the background color of every second row in a CopyFrom table is set to this color.<br>The background color of every other row is as specified on the background color property of the Table.<br>When AlternatingBackColor is specified for both TableRow and TableCell a chequered effect is achieved. |
| TableCell | AlternatingBackColor | When specified, the background color of every second column in a CopyFrom table is set to this color.<br>The background color of every other column is as specified on the background color property of the Table.<br>When AlternatingBackColor is specified for both TableRow and TableCell a chequered effect is achieved. |

## 2.8 Control Specifications, Remove

A Control Specification that has previously been added to a parent in the hierarchy can be removed at any level in the hierarchy by inserting a Control Specification Remove item.

For example, if a specification of a TextBoxControl has been added at the application level, it may be desirable to generate a specific field as a default TextBoxControl. In this case, a Control Specification Remove item can be added to the field to remove the specification for the TextBoxControl.

The properties window of Control Specification Remove is illustrated below.




**ControlName**: This specifies the name of a Standard Control or Custom Control that previously has been specified at a parent level in the hierarchy. This is selected from the dropdown list, which is automatically filled with a list of available Standard Controls and Custom Controls. For convenience, a 'Remove All' item is available in the list. When selecting 'Remove All', all control specifications inherited from the parent levels will be removed.

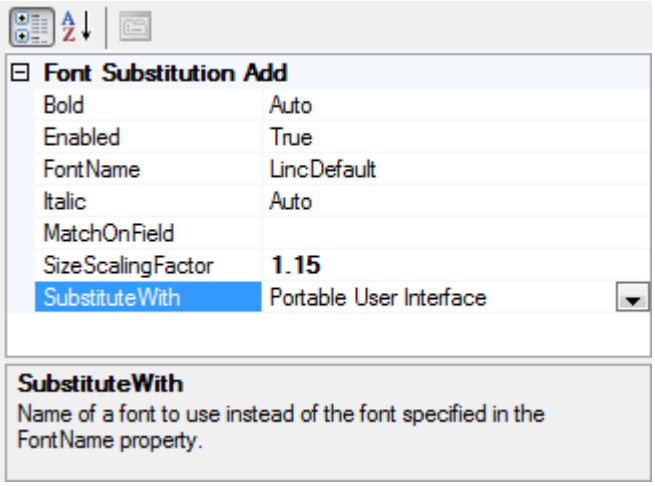
## 2.9 Font Substitutions, Add

Font Substitutions are used when replacing/substituting fonts that are specified at the time of painting a form, with one of the fonts supported by WPF.

Font Substitutions can be added to any level in the hierarchy.

When font substitutions are inherited from parent levels, font substitutions at the current level are evaluated first. When substituting the same font with different fonts specifying match conditions, the substitutions will be evaluated in the order they are specified, except for unconditional substitutions which will be evaluated last.

When adding a Font Substitution, the properties illustrated below are available.




**Bold**: This specifies whether to bold the font. The following selections are available:

- Auto – Use the bold setting as specified on the original font.
- On – Bold the font.
- Off – Do not bold the font.

**Enabled**: This property provides a convenient way of removing a font substitution from the generate process without deleting it.

**FontName**: This specifies the name of the Font to be substituted. This can be selected from the dropdown list of fonts or typed directly.

**Italic**: specifies whether to use italic font. The following selections are available:

- Auto – Use the italic setting as specified on the original font.
- On – Italic the font.
- Off – Do not italic the font.

**SizeScalingFactor**: This property specifies the scaling, which is used for increasing/decreasing the font size. When substituting fonts, it may be necessary to increase/decrease the painted font size to get a better match with the substitute font.

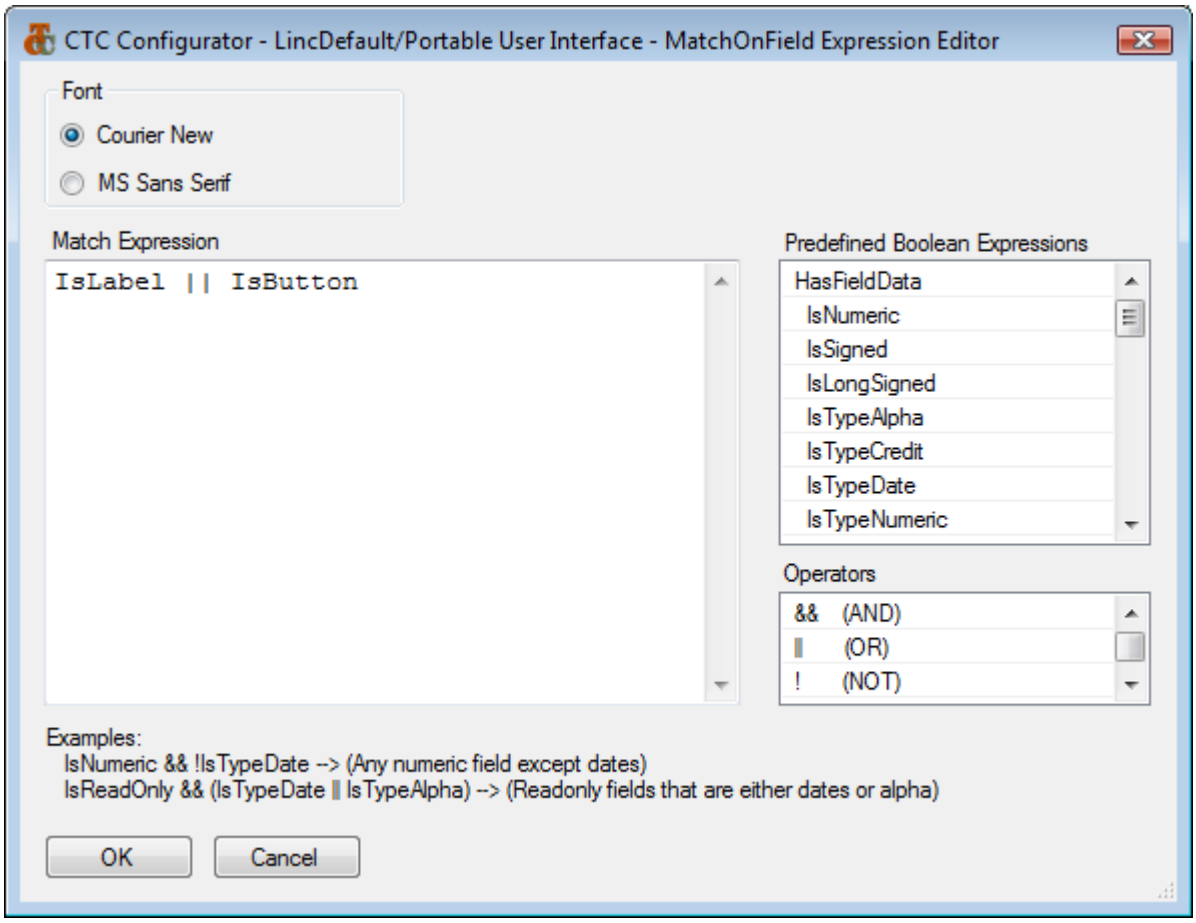
**SubstituteWith**: This specifies the name of a Font to use instead. This can be selected from the dropdown list of fonts supported by WPF or typed directly.

**MatchOnField**: Optional match expression. MatchOnField is a boolean expression specifying the condition for selecting to which field or fields to apply the substitution. At generate time, the MatchOnField expression is evaluated against the field associated with the particular font

and when the expression evaluates True, the substitution is applied. When no expression is specified, control substitution is unconditionally applied.

The expression is specified by selecting the ellipsis button [...]. This will open the Expression Editor window as illustrated below.




An expression is specified using a selection of predefined boolean expressions. Expressions are dragged/dropped or copied/pasted from the list of Predefined Boolean Expressions. Operators such as && (AND), || (OR), ! (NOT), == (Equal), != (Not Equal), > (Greater Than), < (Less Than), >= (Greater Than or Equal) or <= (Less Than or Equal) as well as parentheses ( ) can be used to create complex expressions.

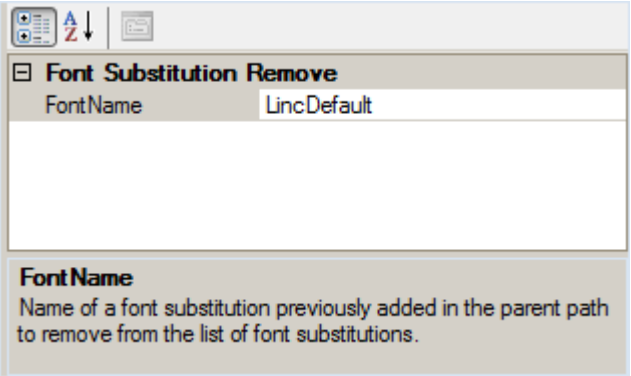
## 2.10 Font Substitutions, Remove

A Font Substitution that has previously been added to a parent in the hierarchy can be removed at any level in the hierarchy by inserting a Font Substitution Remove item.

For example, if a substitution of 'Arial' with 'Courier New' has been added at the application level, it may be desirable to generate a specific field using the painted 'Arial' font and not as 'Courier New'. In this case, a Font Substitution Remove item can be added to the field to remove the substitution for the 'Arial' font.

The properties window of Font Substitution Remove is illustrated below.




**FontName**: This specifies the name of a font that previously has been substituted at a parent level in the hierarchy. This can be selected from the dropdown list of fonts or typed directly. For convenience, a 'Remove All' item is available in the list. When selecting 'Remove All', all font substitutions inherited from the parent levels will be removed.

## 2.11 Options

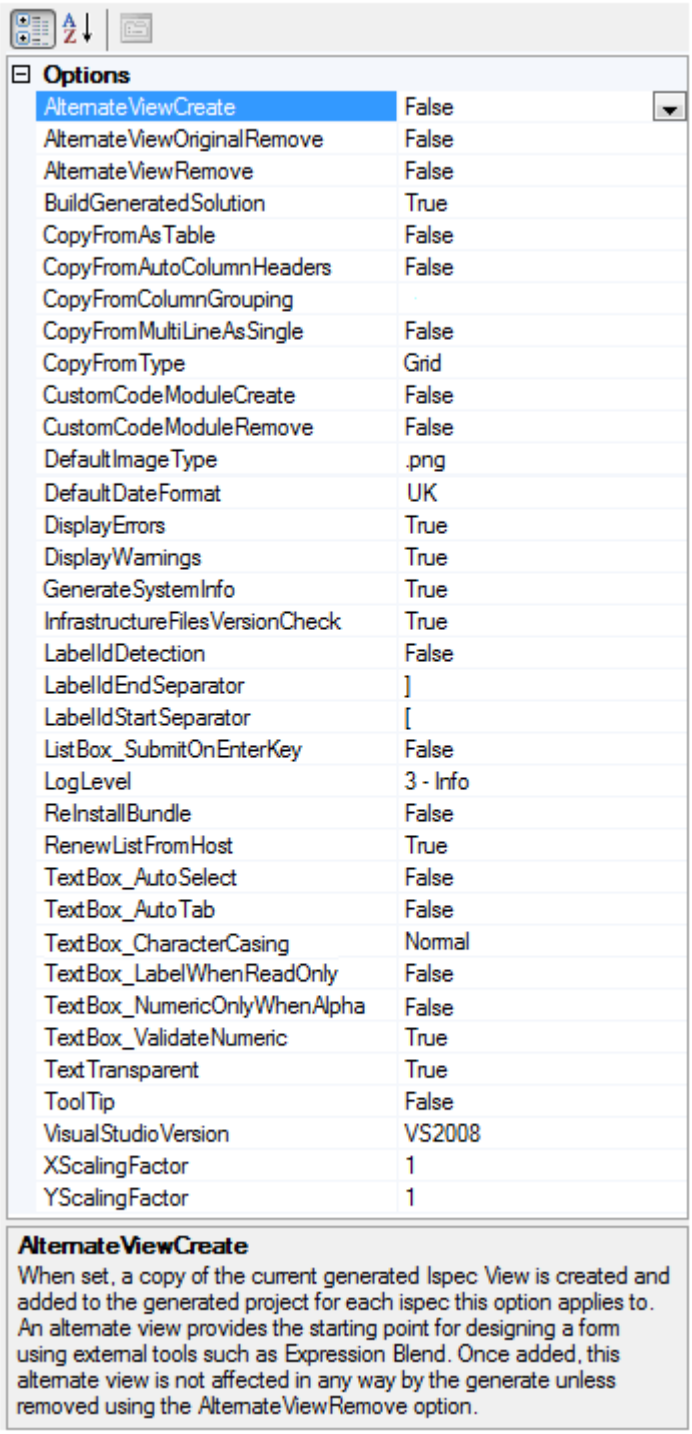
Options specify various settings that take effect at generate time.

Options can be specified at any level in the hierarchy.

When selecting an option from the tree structure, for convenience the complete list of options for that level will be shown in the properties window highlighting the option selected. This allows editing any or all of the options in the properties window without having to select each option individually from the tree structure.

A sample options properties window is illustrated below including default values.




The table below lists the options available for the CTC WPF Client Generator.

| Option | Description |
|---|---|
| AlternateViewCreate | When set, a copy of the current generated Ispec View is created and added to the generated project for each ispec this option applies to.<br><br>Creating an alternate view from the generated Ispec View provides an easy way to get started on designing a form using external tools such as Expression Blend.<br><br>Once added, this alternate view is not affected in any |

| | |
|---|---|
| | way by the generate unless removed using the AlternateViewRemove option. |
| AlternateViewOriginalRemove | When set and an Alternate View exist for an ispec, the original Ispec View will be removed from the generated project. Removing the original Ispec View from the project keeps the size of the package to be downloaded to the client to a minimum. |
| AlternateViewRemove | When set, the reference to the alternate view is removed from the generated project for each ispec this option applies to. The alternate view files are not deleted. |
| BuildGeneratedSolution | Automatically compile the generated application and build the required dll's. |
| CopyFromAsTable | When set, the CopyFrom region painted on a form is generated as a Table with one row for each CopyFrom copy and each control placed within the appropriate cell. |
| CopyFromAutoColumnHeaders | 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. |
| CopyFromColumnGrouping | Allows specifying grouping of fields in a CopyFrom row into columns.<br>A specification of [1-3,8][4][5-7] specifies three column where fields 1, 2, 3 and 8 go into column 1, field 4 go into column 2 and fields 5, 6, and 7 go into column 3.<br>The CopyFromColumnGrouping option is only meaningful on a CopyFrom ispec. |
| CopyFromMultiLineAsSingle | When set, the generator will generate a CopyFrom spanning multiple lines as single lines. This option may be useful for MultiLine CopyFrom areas that are not suitable as table or grid layout. |
| CopyFromType | Specifies the type of CopyFrom table to generate.<br>'Table' specifies to generate the CopyFrom region as a traditional table based on the WPF Grid Layout control.<br>'Grid' specifies to generate 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 CopyForm table. |
| CustomCodeModuleCreate | When set, an empty code-behind module is created and added to the generated project for each ispec form this option applies to. A custom code module allows the addition of custom code to an ispec form. Once added, this module is not affected in any way by the generator unless specifically removed using the |

| | |
|---|---|
| | CustomCodeModuleRemove option. |
| CustomCodeModuleRemove | When set, the reference to the custom code-behind module is removed from the generated project for each ispec form to which this option applies. The custom code module file is not deleted. |
| DefaultImageType | The file extension to be applied to the image name when the image name returned from the host system doesn't contain a valid extension.<br>The WPF valid image extensions are: JPG and PNG. |
| DefaultDateFormat | The default date format to apply to Date Controls.<br>Possible formats are:<br>• UK (ddMMyy, ddMMyyyy)<br>• US (MMddyy, MMddyyyy)<br>• International (yyMMdd, yyyyMMdd) |
| DisplayErrors | Display errors occurring during the generate process in a message box. |
| DisplayWarnings | Display warnings occurring during the generate process in a message box. |
| GenerateSystemInfo | Writes information about ispecs and fields contained in the bundle being generated to the CTCSystemInfo.xml file.<br>This information is used by the configurator to populate the dropdowns on fields, ispecs, bundles and applications with valid selections.<br>When disabled, the information for the current bundle is removed from the xml file on the next generate. |
| IdentifyGroupBox | 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. |
| InfrastructureFilesVersionCheck | Enables version check on infrastructure files. When true, version check is performed while copying infrastructure files when the ReInstallBundle option is set. Only infrastructure files that are newer than the destination files will be copied.<br>When false, no version check is performed and all infrastructure files are unconditionally copied and existing destination files will be overwritten.<br>Upon successful completion of the generate, the option is automatically reset to true by the generator. |
| LabelIdDetection | Detect label identifier specified in the text of the label.<br>Since labels are not associated with data items and therefore they are not individually identified in the |

| | |
|---|---|
| | painter, this provides a convenient way of identifying labels, allowing for configuration of specific label controls on the form.<br><br>For example, "[Head1]Customer Maintenance" where 'Head1' is the name identifying the label allowing this label control to be referred to as Head1 when configuring the label.<br><br>Label identifier must be alphanumeric containing at least one alpha character or _ (underline).<br><br>Label identifiers are automatically removed from the label text. |
| LabelIdEndSeparator | End separator character for identifying labels. |
| LabelIdStartSeparator | Start separator character for identifying labels. Any text between Start Separator and End Separator within label text identifies a label. |
| ListBox_SubmitOnEnterKey | Allows submission of the form to the server when hitting the enter key on an item in a listbox. |
| LogLevel | Level of details to write to the log file. |
| PositionLeftAdjustment | 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. |
| PositionTopAdjustment | 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. |
| ReInstallBundle | Re-install the bundle by copying infrastructure files to the bundle views directory. This is required when new infrastructure files have been added or to repair damaged files. The re-install is performed when next starting the generate of this bundle. Upon successful completion of the generate, the option is automatically reset by the generator. |
| RemoveButtonGroupPanel | Removes the panel which is added by AB Suite as a group container for Button Groups when importing a model from EAE. |
| Single Solution File | This specifies whether to create a single solution file containing references to all IspecView projects or to create each of the IspecView projects as individual projects and keep the main application solution file as small as possible.<br><br>When the bundle contains about 20 or more ispecs, it can take a long time for Visual Studio to open the solution and when that is the case, it is recommended |

| | |
|---|---|
| | to set this option to false. |
| TextBox_AutoSelect | Automatically highlight the text of a TextBox when the TextBox receives focus. |
| TextBox_AutoTab | 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_CharacterCasing | Specifies the character casing (Upper, Lower or Normal) to apply to a TextBox for alphanumeric fields when typing into the TextBox. |
| TextBox_LabelWhenReadOnly | Indicates whether to generate a Label control instead of a TextBox when the data field is read-only and defined as usage inquiry. |
| TextBox_NumericOnlyWhenAlpha | Ensure the value entered into a textbox is numeric for fields defined as alpha. |
| TextBox_ValidateNumeric | Ensure the value entered into a textbox is numeric for fields defined as numeric. |
| TextTransparent | Ensure controls such as Label and Panel are generated with transparent background. |
| ToolTip | Display the Help Text, when defined for the field, as a tool tip on the control. |
| TwoDigitYearCutoff | This option specifies an integer from 1 to 9999 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 Silverlight DatePicker.<br>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 precedes the cutoff year. For example, if two digit year cutoff is 2056 (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.<br>When a date is entered using the Silverlight DatePicker control without specifying the century, only years in the range 1957-2029 can be determined by the system as valid years. The reason is that the century is determined internally by the DatePicker control using the windows system default TwoDigitYearCutoff, which is 2029, before the system performs the validation. |
| VisualStudioVersion | This option specifies the Visual Studio version to |

| | |
|---|---|
| | generate the bundle projects and solution for such as VS 2008 and VS 2010.<br>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. |
| XScalingFactor | X-axis scaling factor to apply for increasing/reducing the left position and width of controls. |
| YScalingFactor | Y-axis scaling factor to apply for increasing/reducing the top position and height of controls. |

## 2.12 Runtime Configuration

Runtime Configuration specifies parameters to take effect at run time for the generated user interface application.

Runtime Configuration can be specified at bundle level only.

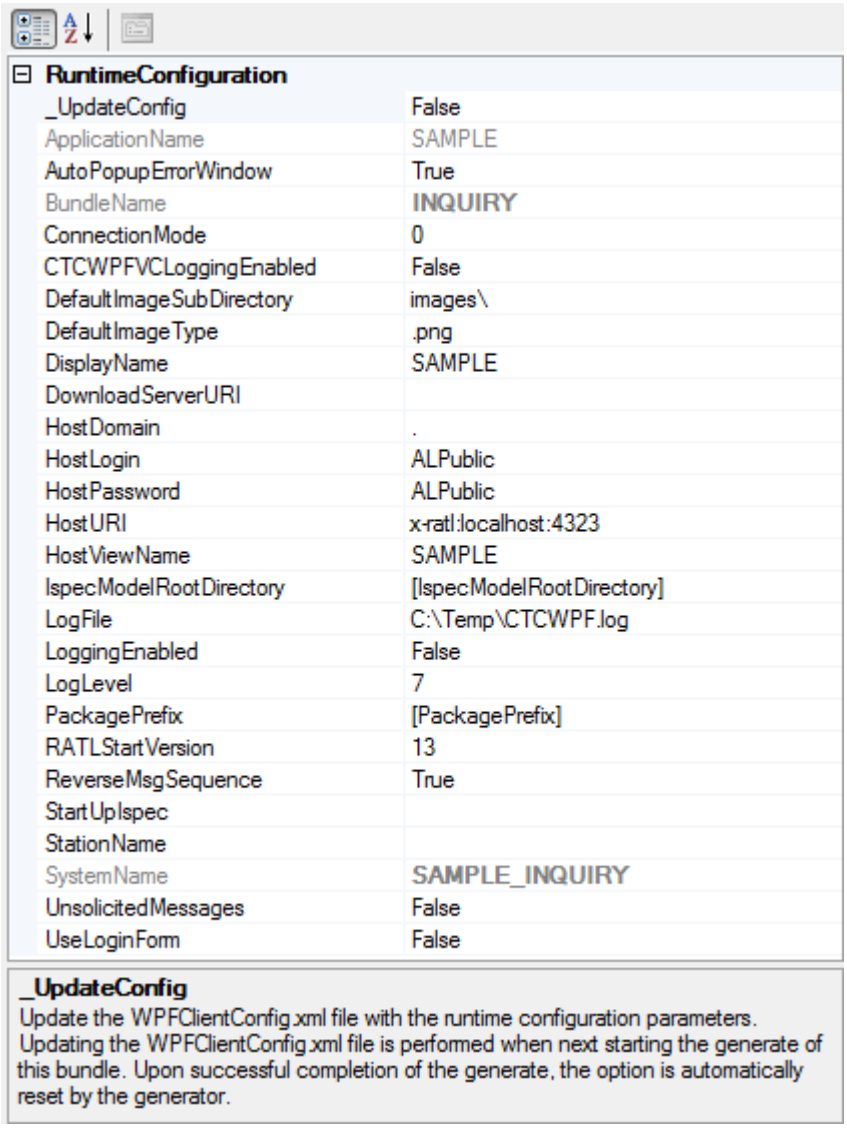
In order for the generator to update the WPFClientConfig.xml file only when the configuration parameters have changed, the _UpdateConfig parameter is automatically set to true when a parameter is changed. However, _UpdateConfig can be set false in case it is not convenient to update the WPFClientConfig.xml file on the next generate.

At generate time, the Runtime Configuration parameters will be written to the WPFClientConfig.xml file when the _UpdateConfig option is set. Any existing parameters will be overwritten.

_UpdateConfig parameter will automatically be set false by the generator when the WPFClientConfig.xml file has been updated. Because the generator updates the configuration file, it is recommended to close the CTC Configurator while running the generator.

A sample Runtime Configuration properties window is illustrated below.




For information on the Runtime Configuration parameters, refer to the relevant section in the documentation for Component Enabler.