# CTC Silverlight Client Generator

Version 2.0.2

Client Tools Consultancy

## Table of Contents

# 1 Introduction

## 1.1 What is CTC Silverlight Client Generator?

The CTC Silverlight Client Generator is a tool for creating a Silverlight 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 Silverlight 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 Silverlight 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 Silverlight 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 Silverlight 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 Silverlight 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 Silverlight 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 Silverlight 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. |

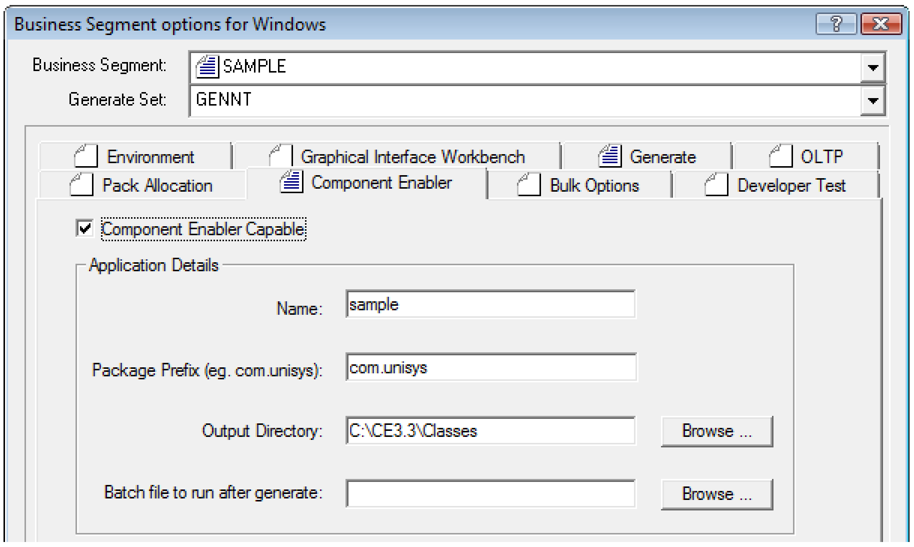
| | |
|---|---|
| TeachText | Text for the teach form. |
| TextBox | Text input and Password box. |

## 2 Generator Initial Setup

The CTC Silverlight 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.


Business Segment options for Windows

Business Segment: SAMPLE
Generate Set: GENNT

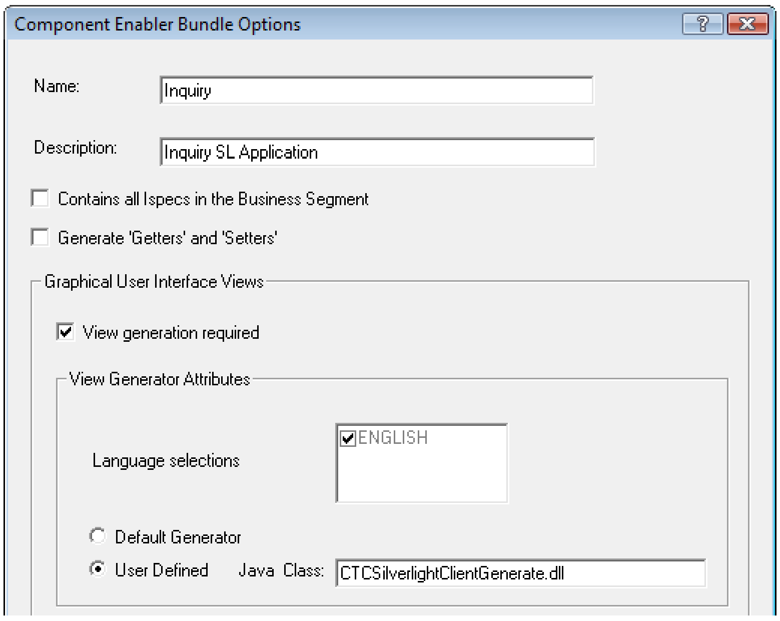
[✓] Component Enabler Capable

Application Details

Name: sample
Package Prefix (eg. com.unisys): com.unisys
Output Directory: C:\CE3.3\Classes
Batch file to run after generate:

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.


Component Enabler Bundle Options
?
Graphical User Interface Views
View Generator Attributes

Name: Inquiry

Description: Inquiry SL Application

☐ Contains all Ispecs in the Business Segment

☐ Generate 'Getters' and 'Setters'

☑ View generation required

Language selections: ☑ ENGLISH

○ Default Generator

● User Defined Java Class: CTCSilverlightClientGenerate.dll

The name of the generator to use must be entered in the Java Class field. The name of the CTC Silverlight 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.

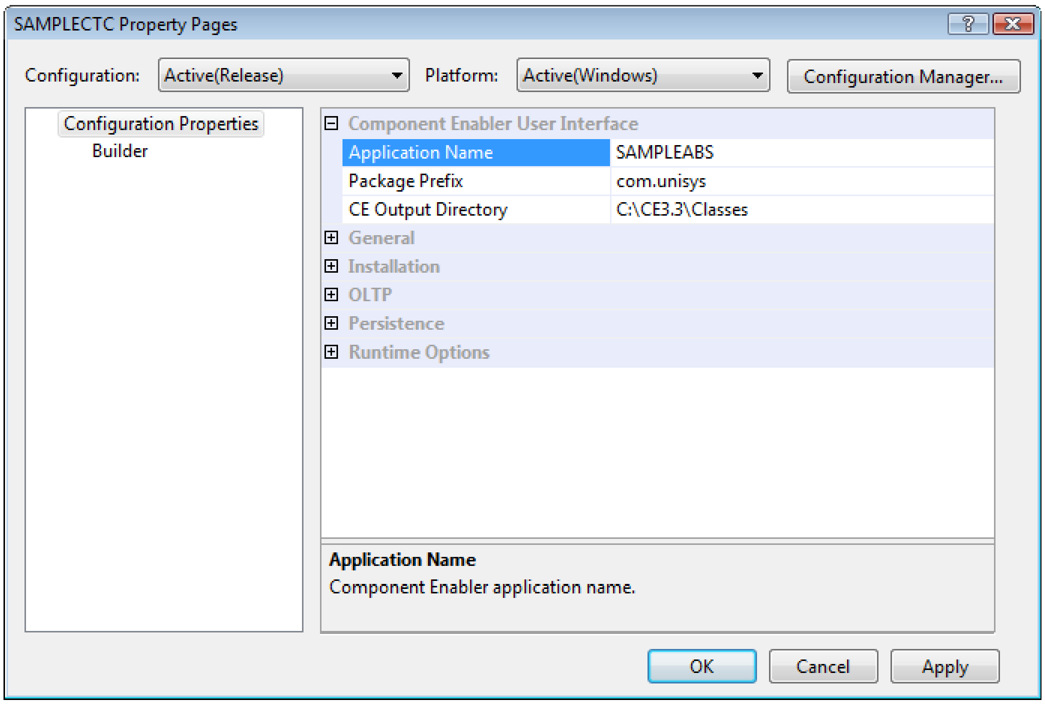
In addition to installing the CTC Silverlight Client Generator, users of EAE 3.3 IC3260 or earlier must also install the CTC Generate Gateway. The CTC Generate Gateway provides the necessary interface allowing the EAE Generate Environment to invoke generators implemented in .NET. Users of EAE 3.3 IC3270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

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

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

## 2.2 AB Suite Setup

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

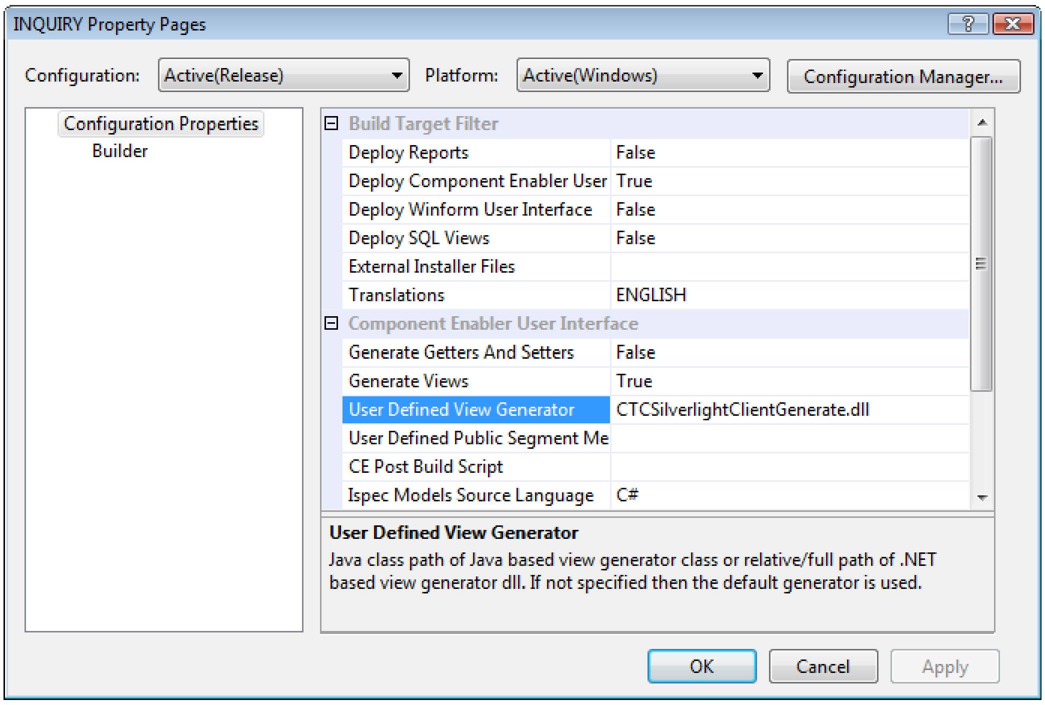


| | |
|---|---|
| Component Enabler User Interface | |
| Application Name | SAMPLEABS |
| Package Prefix | com.unisys |
| CE Output Directory | C:\CE3.3\Classes |
| General | |
| Installation | |
| OLTP | |
| Persistence | |
| Runtime Options | |

**Application Name**
Component Enabler application name.

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.



| | |
|---|---|
| Build Target Filter | |
| Deploy Reports | False |
| Deploy Component Enabler User | True |
| Deploy Winform User Interface | False |
| Deploy SQL Views | False |
| External Installer Files | |
| Translations | ENGLISH |
| Component Enabler User Interface | |
| Generate Getters And Setters | False |
| Generate Views | True |
| User Defined View Generator | CTCSilverlightClientGenerate.dll |
| User Defined Public Segment Me | |
| CE Post Build Script | |
| Ispec Models Source Language | C# |

**User Defined View Generator**
Java class path of Java based view generator class or relative/full path of .NET based view generator dll. If not specified then the default generator is used.

Deploy Component Enabler User Interface must be set to True.

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

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

Ispec Model Source Language must be set to C#.

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

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

## 2.3 Installing Bundle Infrastructure Files

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

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

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

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

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

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

# 3 Configuring the Generator

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

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

For further details on how to use the CTC Configurator, refer to the **CTC Configurator Framework** and **CTC Silverlight 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 Silverlight while Developers concentrate on the back-end implementation.

### Alternate View Original Remove

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 XAP package to be downloaded to the client to a minimum.

### 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 Silverlight.

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 Column Grouping**

This option allows specifying grouping of fields in a CopyFrom row into columns.

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.

The CopyFromColumnGrouping option is only meaningful on a CopyFrom ispec.

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

**CopyFrom MultiLine As Single**

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

**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 Silverlight Grid Layout control.

The 'Grid' option generates the CopyFrom region as a Silverlight 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 Font**

This defines the font to apply when the font on the control is not one of the Silverlight supported fonts and a font substitution is not configured.

The Silverlight supported fonts are: Arial, Arial Black, Comic Sans MS, Courier New, Georgia, Lucida Grande/Lucida Sans Unicode, Times New Roman, Trebuchet MS, Verdana

**Default Image Type**

This defines the file extension to apply to image names when the image name returned from the host system doesn’t contain a valid extension.

The Silverlight valid image extensions are: JPG and PNG.

**Default Date Format**

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

Possible formats are:

- UK (ddMMyy, ddMMyyyy)
- US (MMddyy, MMddyyyy)
- International (yyMMdd, yyyyMMdd)

**Display Errors, Display Warnings**

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

**Exclude Teach Screen**

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

**Generate System Info**

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

**Identify GroupBox**

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

**IIS Reset**

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

**Infrastructure Files Version Check**

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

**Label Id Detection, LabelIdStartSeparator, LabelIdEndSeparator**

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

**ListBox Submit On Double Click**

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

**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 ListBox. DataGrid does not support Submit on Enter Key.

**Log Level**

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

**Out Of Browser**

Enables the Silverlight Out-Of-Browser feature. When set, the Silverlight application generated for the bundle is enabled to run out of the browser on the end user's workstation, providing a desktop look and feel to the application.

When enabled, an option "Install Application to Desktop..." on the main menu of the application allows the end user to install the application to the desktop. The installation process is automatic.

A Silverlight application running out-of-browser is still able to communicate as normal with the backend EAE or AB Suite host application. Updates to the generated Silverlight application are detected and installed automatically.

**Position Left Adjustment**

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

**Position Top Adjustment**

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

**Re-Install Bundle**

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

**Remove Button Group Panel**

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

**Silverlight Version**

This option specifies whether to use 'Silverlight 3' or 'Silverlight 4' when generating a bundle.

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

**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.

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 Auto Select**

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

**TextBox Auto Tab**

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

**TextBox Character Casing**

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

**TextBox Label When Read-Only**

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

**TextBox Numeric Only When Alpha**

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

**TextBox Validate Numeric**

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

**Text Transparent**

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

**Tool Tip**

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

**Two Digit Year Cutoff**

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.

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.

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.

**Virtual Directory Auto Create**

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

**Virtual Directory Name New**

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

**Visual Studio Version**

This option specifies the Visual Studio version to generate the bundle projects and solution for. The value can be VS2008 or VS2010 and it is determined by setting the Silverlight Version property.

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

**X Scaling Factor**

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

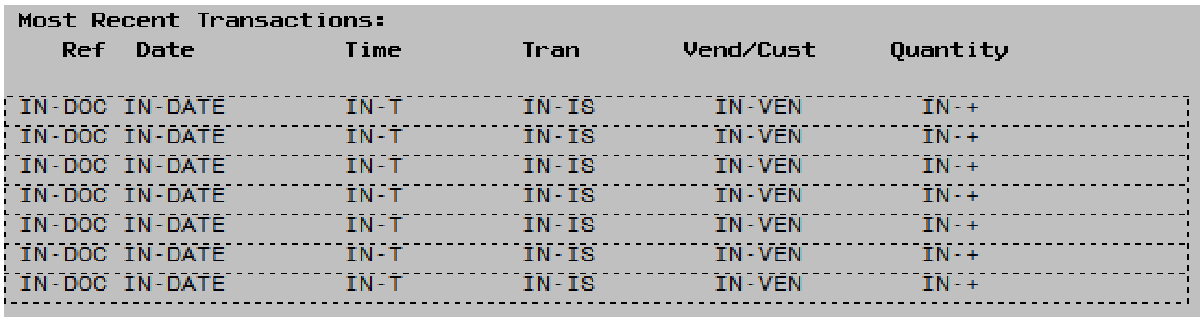
**Y Scaling Factor**

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

Other options are available specific to individual controls. Refer to the **CTC Silverlight Client Configurator** documentation for further details.

## 3.2 Generating CopyFrom As Table

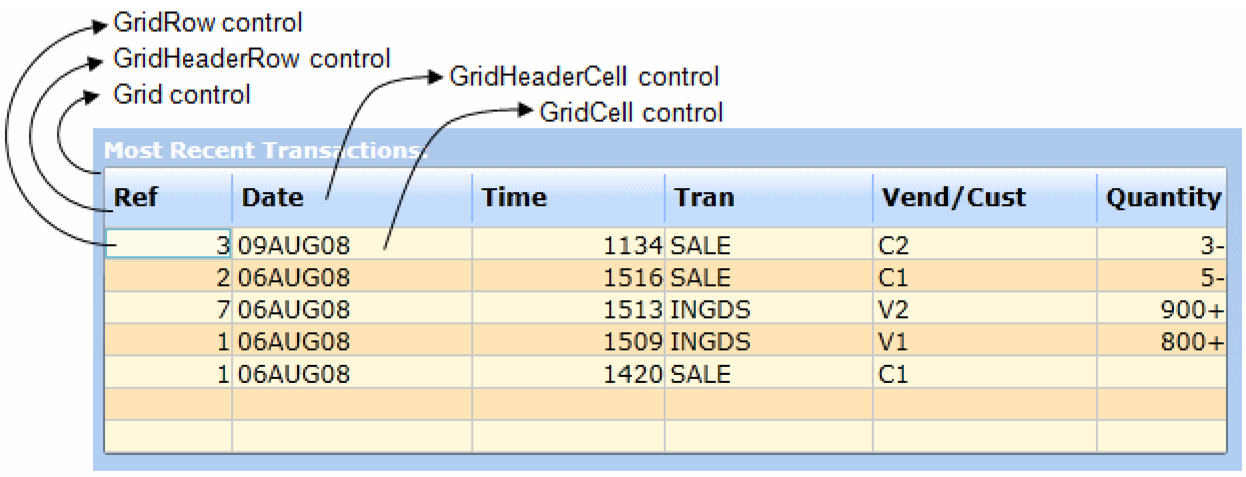
As an example, when setting the options CopyFromAsTable and CopyFromAutoColumnHeaders, a CopyFrom region painted as the following in the EAE or AB Suite development environment:



```
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-+
```

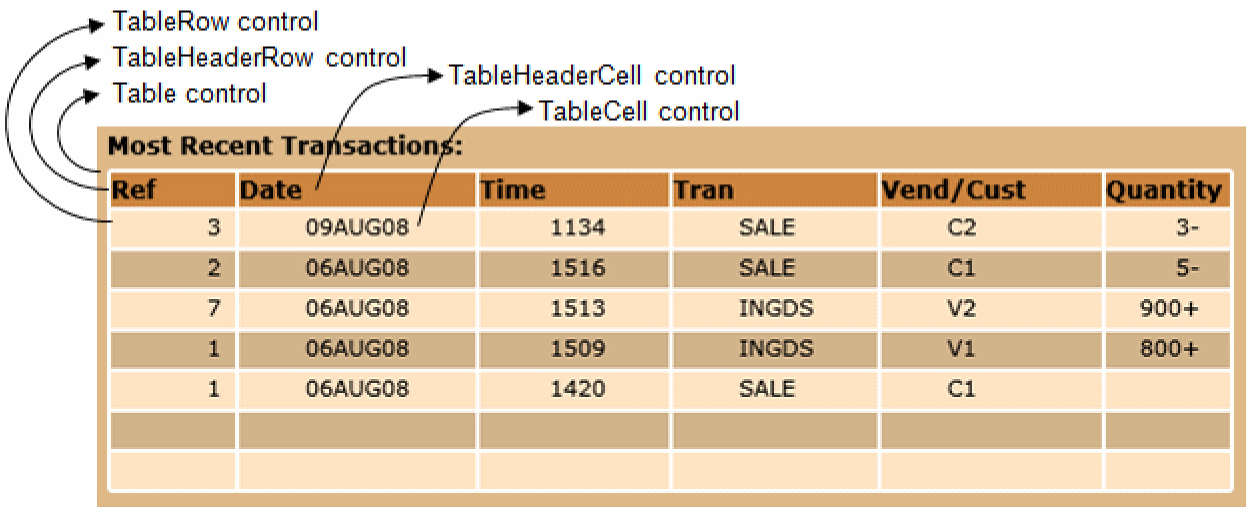
is generated as follows:



Most Recent Transactions

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

The table above is an example of generating the CopyFrom region with the option CopyFromType = "Grid". The table is generated as a Silverlight 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.



Most Recent Transactions:

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

The table above is an example of generating the CopyFrom region with the option CopyFromType = 'Table'. The table is generated as a traditional Silverlight 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 Silverlight 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. It is also assumed that a label has been painted for each column/field in the CopyFrom, including columns containing hidden fields, and columns containing multiple fields. It may therefore be necessary to add dummy (empty) label controls to the painted forms. When using AB Suite, and label controls are painted inside a panel, they will not be discovered. The label controls are placed in the TableHeaderCells in the order they are painted.

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

1. Label controls that are painted as column headers can be identified to the generator for the purpose of being used as column headers. As part of the label text, a column identifier can be specified as: "Date[02]". The generator will recognize the notation [02] and use this as the column number.

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

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

2. Using the CTC Configurator, the individual 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.

Fields in a CopyFrom row can be grouped and placed in the same column by using the CopyFromColumnGrouping option or by specifying grouping on the painted label controls.

- CopyFromColumnGrouping option:
  For example, specifying CopyFromColumnGrouping = "[1-3][4][5-7]" creates three columns where fields 1, 2 and 3 are grouped into column 1, field 4 into column 2 and fields 5, 6, and 7 into column 3.

- Label control option:
  "Account Number[01(1-3)]" specifies fields 1, 2 and 3 to be grouped into column 1 with "Account Number" in the column header.
  "Date[02(4)]" specifies field 4 to go into column 2 with "Date" in the column header.

"Additional Info[03(5-7)]" specifies fields 5, 6 and 7 to be grouped into column 3 with "Additional Info" in the column header.

Column numbers must be consecutive starting with 1.

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

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

Column grouping allows moving fields into columns in an order that is different to the order in which they are painted. However, moving fields in the same sequence as they are painted, gives the generator the best conditions for calculating the width of columns as close to the painted form as possible.

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

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

## 3.3 Runtime Configuration

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

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

## 3.4 Control Specifications

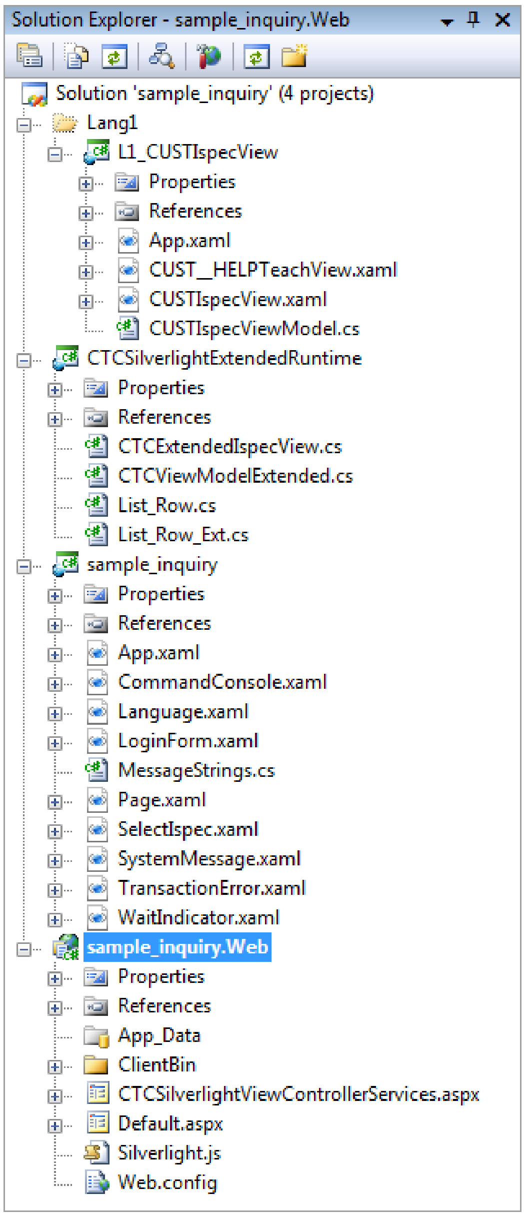
All properties on any Standard or Custom Control can be configured. The visual appearance of any control can be specified via the CTC Configurator. Any property defined by the Microsoft Silverlight controls can be specified on the controls.

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

# 4 The Generated User Interface Application

## 4.1 The Visual Studio Solution

The generated application is a Silverlight application created for Visual Studio 2008. A Silverlight solution containing three 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.


```
Solution Explorer - sample_inquiry.Web
Solution 'sample_inquiry' (4 projects)
  Lang1
    L1_CUSTIspecView
      Properties
      References
      App.xaml
      CUST__HELPTeachView.xaml
      CUSTIspecView.xaml
      CUSTIspecViewModel.cs
  CTCSilverlightExtendedRuntime
    Properties
    References
    CTCExtendedIspecView.cs
    CTCViewModelExtended.cs
    List_Row.cs
    List_Row_Ext.cs
  sample_inquiry
    Properties
    References
    App.xaml
    CommandConsole.xaml
    Language.xaml
    LoginForm.xaml
    MessageStrings.cs
    Page.xaml
    SelectIspec.xaml
    SystemMessage.xaml
    TransactionError.xaml
    WaitIndicator.xaml
  sample_inquiry.Web
    Properties
    References
    App_Data
    ClientBin
    CTCSilverlightViewControllerServices.aspx
    Default.aspx
    Silverlight.js
    Web.config
```

The project named **sample_inquiry** represents the Silverlight client application. It contains fixed non-generated files required for the Silverlight application on the client/browser side. The Page.xaml file is the main page that opens in the browser when the user starts the application. This page hosts the CTC Silverlight 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 Silverlight View Controller, see section below.

The project named **sample_inquiry.Web** represents the web server side of the application. It contains fixed non-generated files required for the application on the server side. The Default.aspx file is the start-up page for the Silverlight application. The URL to start the application points to this file.

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

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

The project named **CTCSilverlightExtendedRuntime** 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 three 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 ispecs only when they are required.

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

- TeachView - This is the teach/help information generated as a Silverlight 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 Silverlight XAML form. Controls on the form bind to properties on the IspecViewModel.

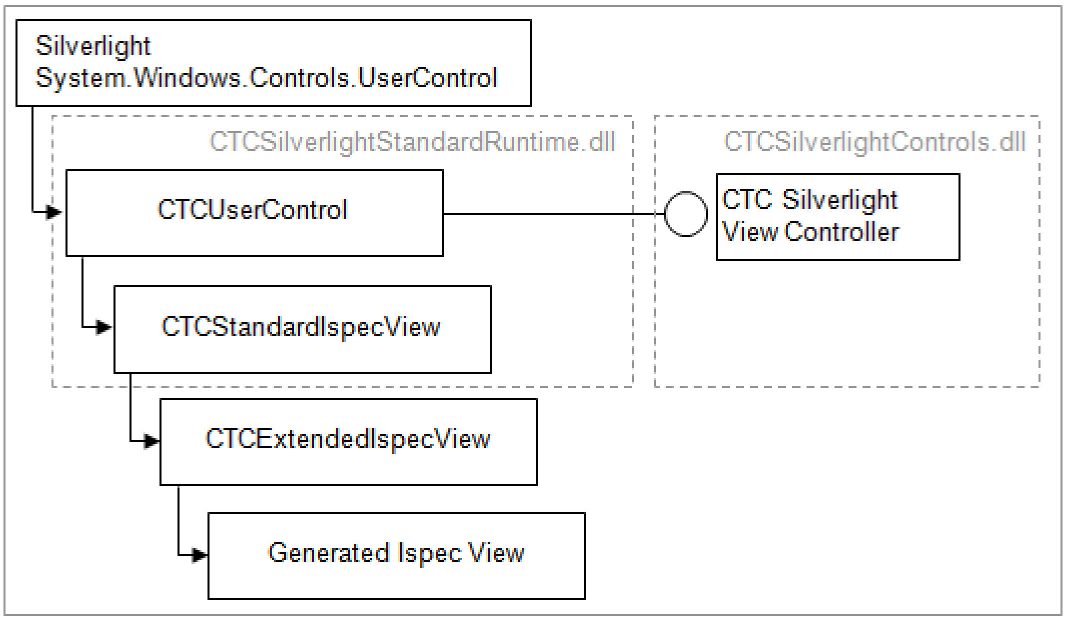
- IspecViewModel – This is a .NET class providing the interface between the controls on the form and the data held in the ispec model. This class contains one property for each data field defined on the ispec.

Ispecs are generated taking full advantage of the Silverlight data binding capabilities. Each control on the form uses the Silverlight 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 Silverlight 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.


Silverlight System.Windows.Controls.UserControl
CTCSilverlightStandardRuntime.dll
CTCSilverlightControls.dll
CTCUserControl
CTC Silverlight View Controller
CTCStandardIspecView
CTCExtendedIspecView
Generated Ispec View

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

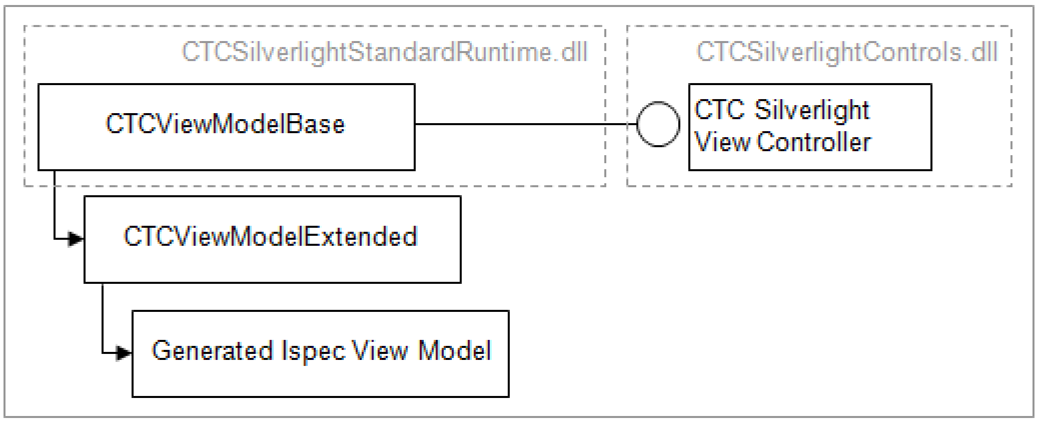
The CTCUserControl and CTCStandardIspecView classes are supplied as a dll named CTCSilverlightStandardRuntime.dll which is installed in the bin directory of the generate solution.

## 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.


CTCSilverlightStandardRuntime.dll
CTCSilverlightControls.dll
CTCViewModelBase
CTC Silverlight View Controller
CTCViewModelExtended
Generated Ispec View Model

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 CTCSilverlightExtendedRuntime, part of the generated project as seen in section 4.1.

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

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

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

## 4.4 Printing

With Silverlight 4, printing data from an ispec specially formatted for printing purposes can be achieved using the Silverlight Printing API.

To support printing, the CTC Silverlight solution provides the capability to include templates created specifically for formatting the data for an ispec for printing purposes with the XAP package of the Ispec View. This utilizes the existing mechanism for downloading forms to the end user workstations.

### 4.4.1 General

The support for custom printing is provided by adding a Custom Code Module to all ispecs for which printing is required. Within the Custom Code Module, the following is available for the user to implement custom printing:

- **CanPrintIspecView** property: This is a Boolean for indicating whether the ispec can be printed. This property can be used from the main Page.xaml.cs class to determine whether a print button or a menu print option can be enabled.
- **PrintIspecView** method: This method can be called from the main Page.xaml.cs class when the user clicks the print button or a menu print option. The user implements the necessary code to perform the printing within this method.
- **CanPrintPreview** property: This is a Boolean for indicating whether the ispec print can be previewed. This property can be used from the main Page.xaml.cs class to determine whether a print preview button or a menu print preview option can be enabled.
- **PrintPreview** method: This method can be called from the main Page.xaml.cs class when the user clicks the print preview button or a menu print preview option. The user implements the necessary code to perform the print preview within this method utilizing the implementation of the PrintIspecView method.

The CTCStandardIspecView class, from which all IspecViews inherit, provides two methods specifically for support of printing:

- **GetUserControlInstance:** This method gets an instance of a xaml UserControl for printing purposes that has been included with the XAP package of the ispec view. Templates for printing are created as xaml UserControl and included with the Ispec View project to become part of the XAP package for the Ispec View.
- **GetCopyFromList:** This method is specific to CopyFrom ispecs. It gets the CopyFrom rows of the ispec as a list that can be passed into the print template.
  This method gets all CopyFrom rows of the ispec by performing multiple transactions to the ispec on the host to retrieve all rows. When getting all CopyFrom rows, this method assumes that transmitting the ispec to the host will return the next group of rows.
  The condition for detecting end-of-copyfrom used by this method is first to check for a row where all columns are empty, second check for whether the host ispec returned the same group as previously, and last check for whether the host ispec started from the beginning of the CopyFrom.
  When multiple transactions are required for getting all the rows, the method will start from the current position. If the current view of the ispec is not positioned at the first row, the user of this method must set the necessary field(s) on the ispec model to the required value(s) to request the ispec to start at the beginning before calling this method.
  This method does not reset the view of the ispec to the state before the method was invoked.

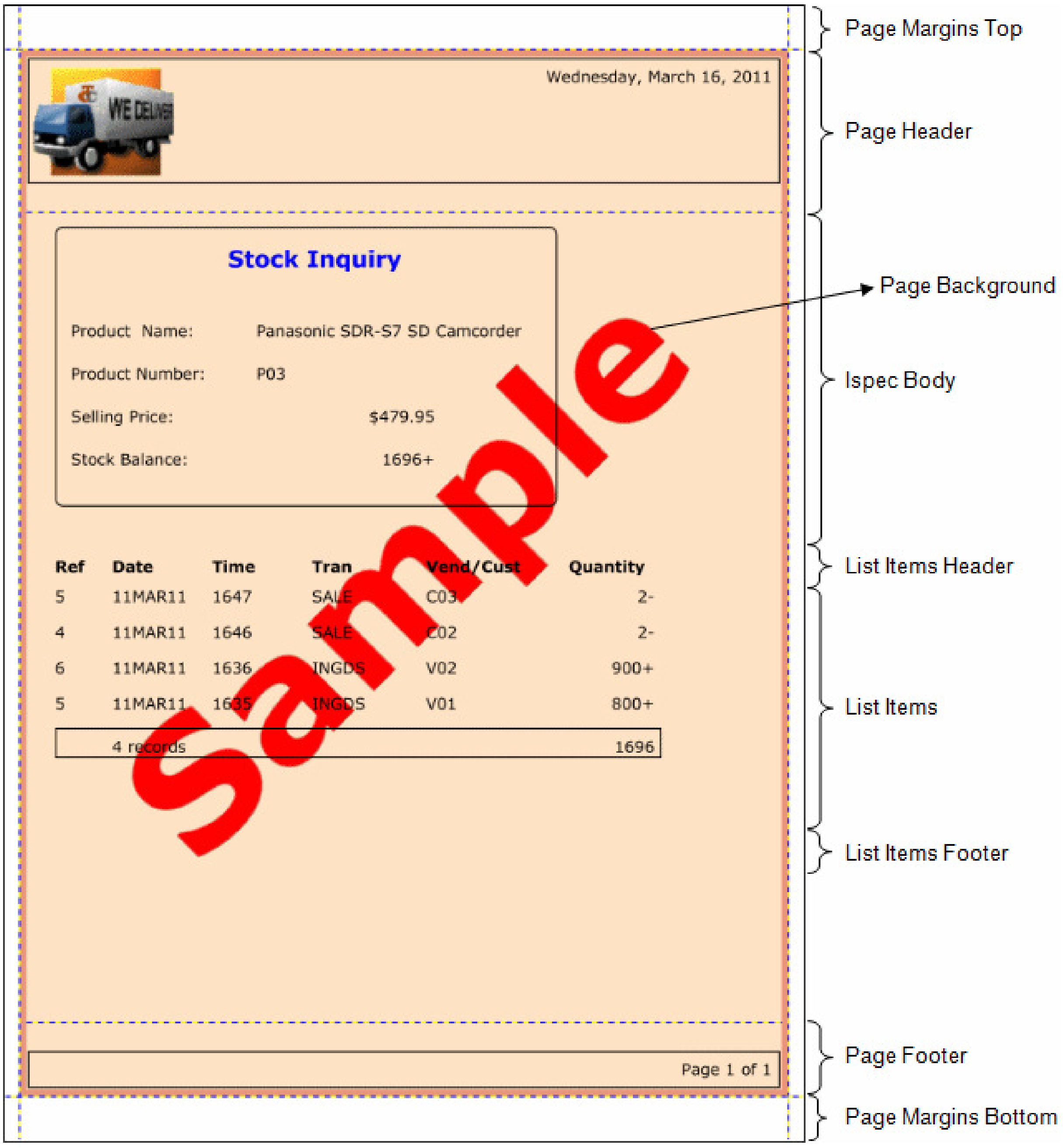
### 4.4.2 Multipage Printing

The Silverlight Printing API can be used directly as is for printing simple single page documents. Printing more complex documents with lists containing a variable number of rows spanning multiple pages requires an approach that measures the space available while adding rows from the list to the page to determine when to create a new page.

The CTC Silverlight **IspecPrint** control is a control that provides support for multipage printing and includes the following features:

- Automatic pagination
- Support for list items of varying height
- Total page count
- Templating
- Page headers and footers
- List header and footer with support for calculated fields
- Multiple Copies
- Events to allow hooking into printing at various stages

The IspecPrint control provides a number of templates for the various sections of a page as shown in the following:



Wednesday, March 16, 2011

**Stock Inquiry**

| | |
|---|---|
| Product Name: | Panasonic SDR-S7 SD Camcorder |
| Product Number: | P03 |
| Selling Price: | $479.95 |
| Stock Balance: | 1696+ |

| Ref | Date | Time | Tran | Vend/Cust | Quantity |
|---|---|---|---|---|---|
| 5 | 11MAR11 | 1647 | SALE | C03 | 2- |
| 4 | 11MAR11 | 1646 | SALE | C02 | 2- |
| 6 | 11MAR11 | 1636 | INGDS | V02 | 900+ |
| 5 | 11MAR11 | 1635 | INGDS | V01 | 800+ |
| | 4 records | | | | 1696 |

Page 1 of 1

The content and layout of each of the templates is user defined. All templates are optional. Only the templates required for the desired printout will need to be specified.

The IspecPrint control provides the following events that allow controlling the printout programmatically:

- BeginPrint – Occurs once when the printing starts. This allows the application to initialize local variables or start a busy indicator.
- EndPrint – Occurs once when the printing has finished. This allows the application to clear local variables or stop a busy indicator.
- BeginBuildPages – Occurs once at the start of building the pages.
- EndBuildPages – Occurs once at the end of building the pages.

- LoadingPageTemplates – Occurs for every new page when templates for the page are being loaded. This allows the application to set or remove templates dynamically for each page.
- PageTemplatesLoaded – Occurs for every new page when the templates have been loaded. This allows the application to set the datacontext for the templates or change the controls within the templates.
- BeginBuildListItem – Occurs for every list item when the item is being added to the page. This allows the application to build running totals.
- BegingBuildListFooter – Occurs once when all list items have been added and starting to build the footer for the list. This allows the application to set running total as datacontext for the list footer.
- SpoolingPageCopy – Occurs for every page when sending a copy of a page to the printer. The event is only raised when the Copies property is set to 2 or higher. This allows the application to change the text on controls within the page. When this event is raised, the page has been arranged and measured, therefore any changes that affect the size of controls may cause the page to overflow and be cut.

### 4.4.3 Examples

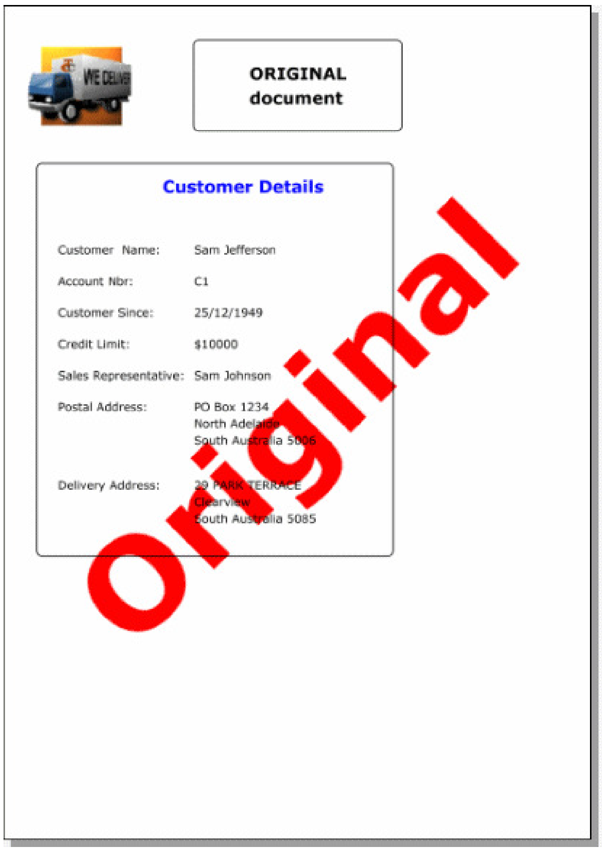
Examples of printing are provided with the installation of the CTC Silverlight Generator. The following examples are available, demonstrating different scenarios for printing:

- Printing of a Standard Component Ispec.
  This example is based on the 'standard' Sample system and is available in
  [ceroot]\CTC-Software\CTC Silverlight Client Generator\views\Lang\CUSTIspecView

  This includes two examples:

  - Example 1 – using the Silverlight Printing API directly. The example shows how to get an instance of a print template UserControl, how to fill the template with data from the ispec using the Data Binding and how to open the print dialog and send the template to the printer.
  - Example 2 – using the CTC IspecPrint control. The example shows how to specify templates for the page layout of a single page printout and printing two copies.
    A sample printout of the CUST ispec is shown below.
- Printing of a CopyFrom Ispec.
  This example is based on the 'standard' Sample system and is available in
  [ceroot]\CTC-Software\CTC Silverlight Client Generator\views\Lang\SINQIspecView

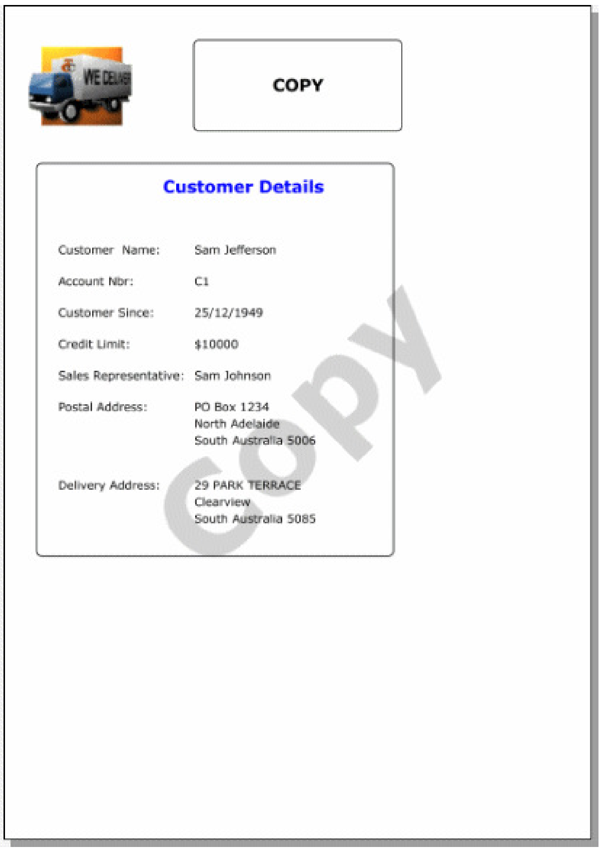
  This example shows how to use the IspecPrint control for printing ispec data including CopyFrom list spanning multiple pages. The example includes page header and footer with page count and total number of pages, CopyFrom list header, list items and footer with running total.
  A sample printout of the SINQ ispec is shown below.



**ORIGINAL document**

**Customer Details**

| | |
|---|---|
| Customer Name: | Sam Jefferson |
| Account Nbr: | C1 |
| Customer Since: | 25/12/1949 |
| Credit Limit: | $10000 |
| Sales Representative: | Sam Johnson |
| Postal Address: | PO Box 1234<br>North Adelaide<br>South Australia 5006 |
| Delivery Address: | 29 PARK TERRACE<br>Clearview<br>South Australia 5085 |

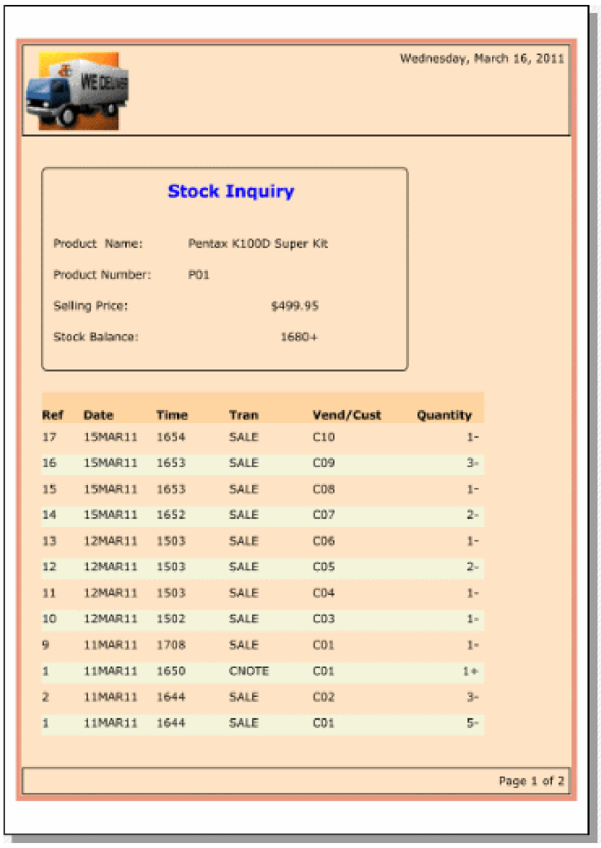


**COPY**

**Customer Details**

| | |
|---|---|
| Customer Name: | Sam Jefferson |
| Account Nbr: | C1 |
| Customer Since: | 25/12/1949 |
| Credit Limit: | $10000 |
| Sales Representative: | Sam Johnson |
| Postal Address: | PO Box 1234<br>North Adelaide<br>South Australia 5006 |
| Delivery Address: | 29 PARK TERRACE<br>Clearview<br>South Australia 5085 |

Sample CUST ispec printout including two copies



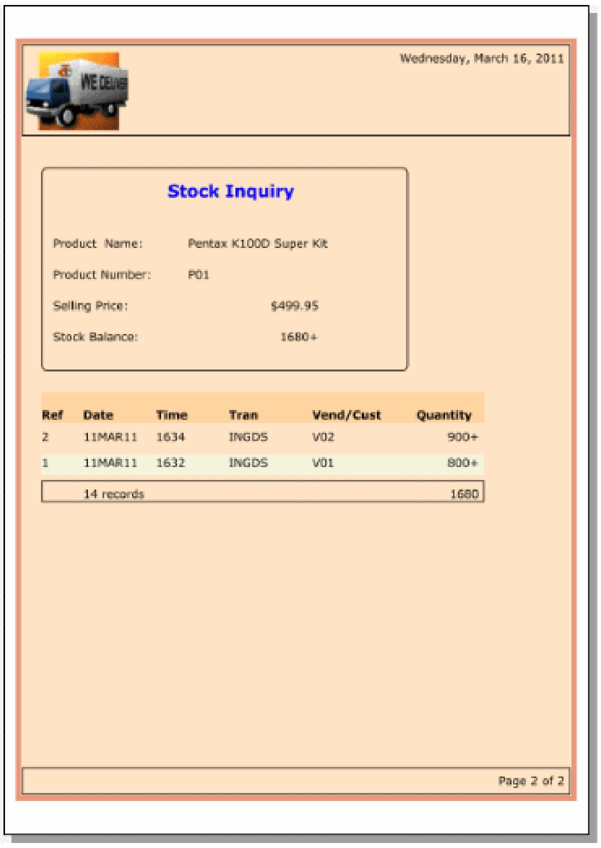
Wednesday, March 16, 2011

**Stock Inquiry**

| | |
|---|---|
| Product Name: | Pentax K100D Super Kit |
| Product Number: | P01 |
| Selling Price: | $499.95 |
| Stock Balance: | 1680+ |

| Ref | Date | Time | Tran | Vend/Cust | Quantity |
|---|---|---|---|---|---|
| 17 | 15MAR11 | 1654 | SALE | C10 | 1- |
| 16 | 15MAR11 | 1653 | SALE | C09 | 3- |
| 15 | 15MAR11 | 1653 | SALE | C08 | 1- |
| 14 | 15MAR11 | 1652 | SALE | C07 | 2- |
| 13 | 12MAR11 | 1503 | SALE | C06 | 1- |
| 12 | 12MAR11 | 1503 | SALE | C05 | 2- |
| 11 | 12MAR11 | 1503 | SALE | C04 | 1- |
| 10 | 12MAR11 | 1502 | SALE | C03 | 1- |
| 9 | 11MAR11 | 1708 | SALE | C01 | 1- |
| 1 | 11MAR11 | 1650 | CNOTE | C01 | 1+ |
| 2 | 11MAR11 | 1644 | SALE | C02 | 3- |
| 1 | 11MAR11 | 1644 | SALE | C01 | 5- |

Page 1 of 2



Wednesday, March 16, 2011

**Stock Inquiry**

| | |
|---|---|
| Product Name: | Pentax K100D Super Kit |
| Product Number: | P01 |
| Selling Price: | $499.95 |
| Stock Balance: | 1680+ |

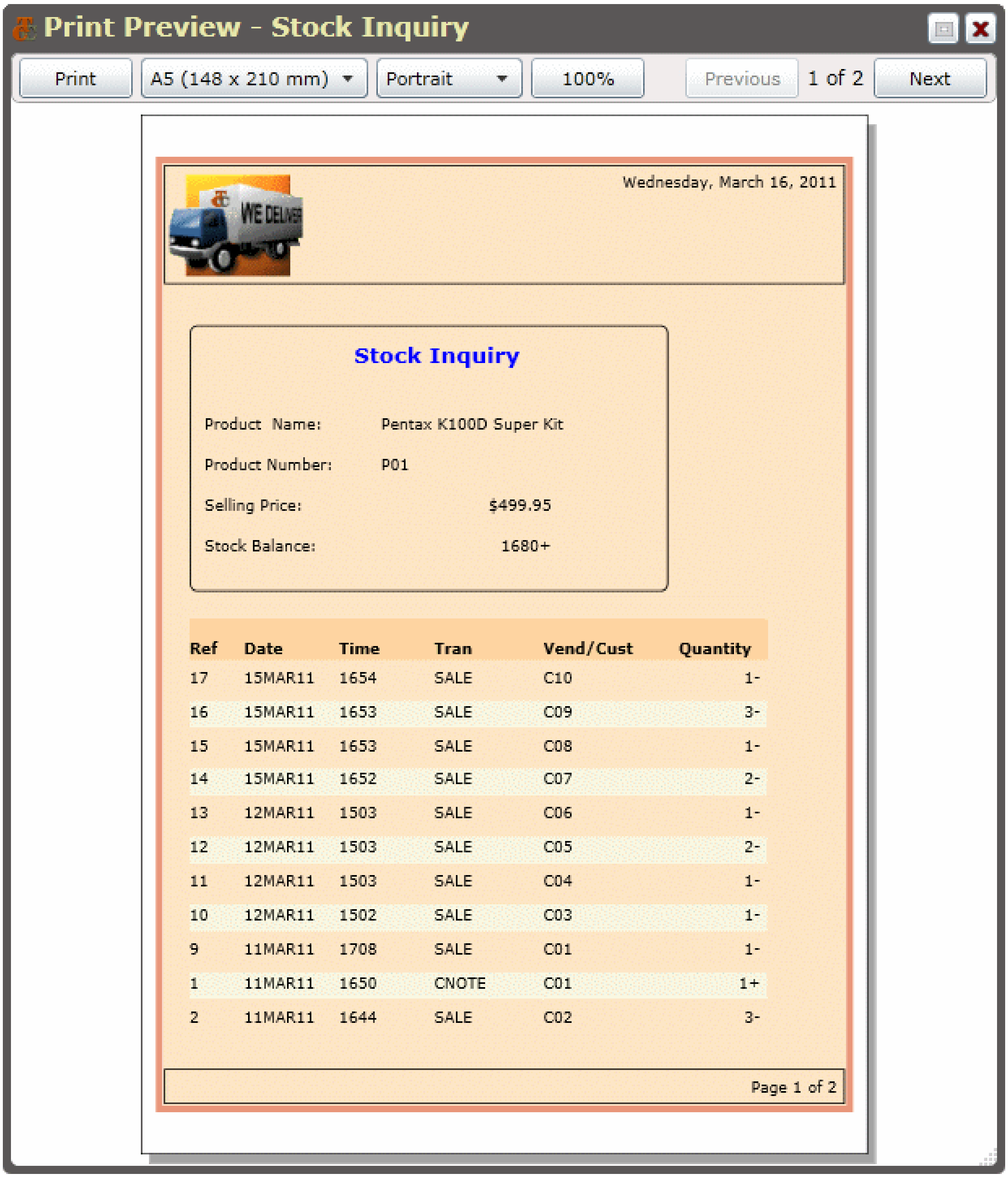
| Ref | Date | Time | Tran | Vend/Cust | Quantity |
|---|---|---|---|---|---|
| 2 | 11MAR11 | 1634 | INGDS | V02 | 900+ |
| 1 | 11MAR11 | 1632 | INGDS | V01 | 800+ |
| | 14 records | | | | 1680 |

Page 2 of 2

Sample SINQ ispec multipage printout

### 4.4.4 Print Preview

Previewing the print of an ispec can be achieved using the Print Preview function provided with the release. The two examples (CUST and SINQ) as mentioned in section 4.4.3 Examples includes a sample implementation of how to add Print Preview. Below is shown an example of the Print Preview function:



Wednesday, March 16, 2011

Stock Inquiry

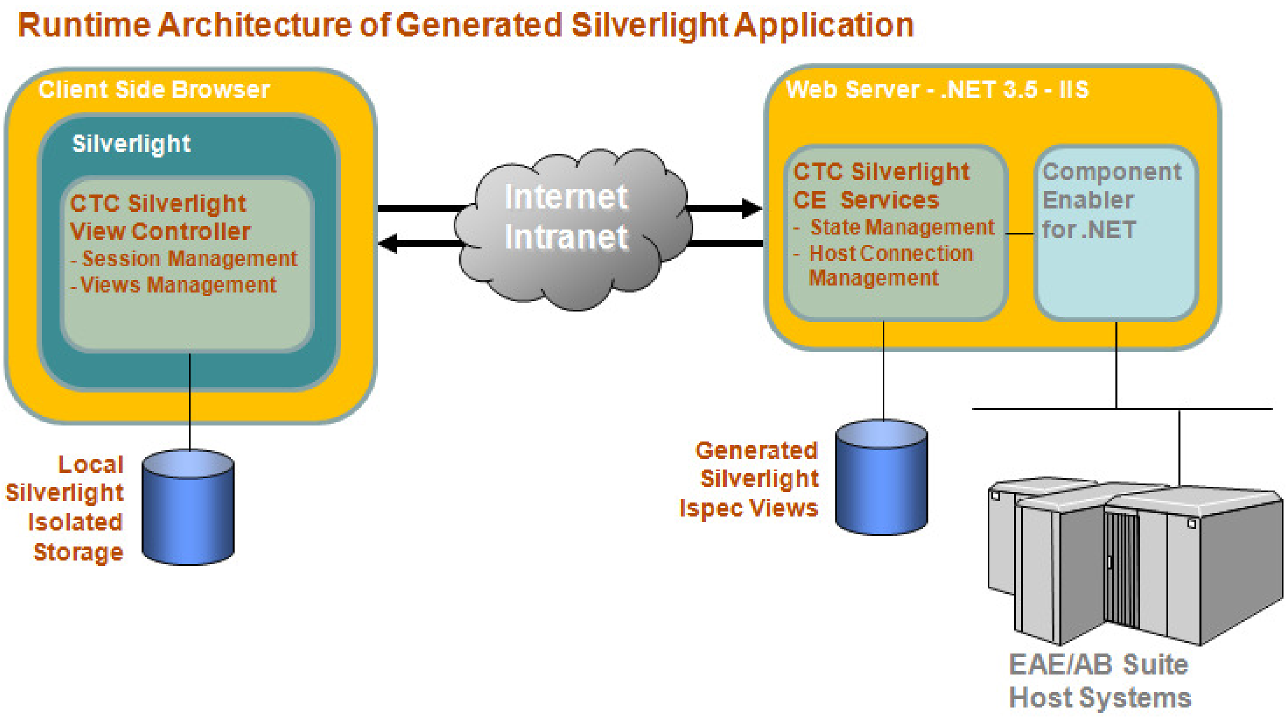
| | |
|---|---|
| Product Name: | Pentax K100D Super Kit |
| Product Number: | P01 |
| Selling Price: | $499.95 |
| Stock Balance: | 1680+ |

| Ref | Date | Time | Tran | Vend/Cust | Quantity |
|---|---|---|---|---|---|
| 17 | 15MAR11 | 1654 | SALE | C10 | 1- |
| 16 | 15MAR11 | 1653 | SALE | C09 | 3- |
| 15 | 15MAR11 | 1653 | SALE | C08 | 1- |
| 14 | 15MAR11 | 1652 | SALE | C07 | 2- |
| 13 | 12MAR11 | 1503 | SALE | C06 | 1- |
| 12 | 12MAR11 | 1503 | SALE | C05 | 2- |
| 11 | 12MAR11 | 1503 | SALE | C04 | 1- |
| 10 | 12MAR11 | 1502 | SALE | C03 | 1- |
| 9 | 11MAR11 | 1708 | SALE | C01 | 1- |
| 1 | 11MAR11 | 1650 | CNOTE | C01 | 1+ |
| 2 | 11MAR11 | 1644 | SALE | C02 | 3- |

Page 1 of 2

## 5 CTC Silverlight View Controller

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


Runtime Architecture of Generated Silverlight Application
Client Side Browser
Silverlight
CTC Silverlight View Controller
- Session Management
- Views Management
Local Silverlight Isolated Storage
Internet Intranet
Web Server - .NET 3.5 - IIS
CTC Silverlight CE Services
- State Management
- Host Connection Management
Component Enabler for .NET
Generated Silverlight Ispec Views
EAE/AB Suite Host Systems

The diagram above provides an overview of the runtime architecture of a Silverlight application generated by the CTC Silverlight 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 Silverlight View Controller manages the session with the host system starting with connecting to the host system, displaying ispec forms/views as the user navigates through the host application, sending and receiving data to and from the host system and ending with disconnecting from the host system. During the interaction with the host system, the View Controller calls upon services delivered by the CTC Silverlight CE Services module running on the web server. The session established with the web server does not timeout, as the View Controller automatically keeps it alive for the duration of the session.

The CTC Silverlight CE Services is implemented as an ASP.NET component. It uses the standard Component Enabler from Unisys to communicate with the EAE/AB Suite host system. The state of a session established by the client View Controller is maintained throughout the session. This allows efficient exchange of data between the server and the client. The amount of data required to be sent between the server and the client is kept to a minimum. By default, the state of connection to the host system is maintained throughout the session. However, the CTC Silverlight CE Services can be configured to use connection object pooling.

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

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

## 5.1 CTC Silverlight View Controller API

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

- Page (xaml + xaml.cs): Main Silverlight UI application installed as the default to the generated bundle.
- PageMultiIspecs (xaml + xaml.cs): Main Silverlight UI application allowing users to keep multiple ispecs open at the same time within one session, including optionally opening multiple copies of the same ispec. Utilizing this requires the EAE or AB Suite applications to be designed and implemented as 'stateless' applications.
- PageMultiSessions (xaml + xaml.cs): Main Silverlight UI application allowing users to open multiple sessions to the EAE and AB Suite host systems. This is to provide users of 'state full' EAE and AB Suite applications a way to let their end users access and view multiple ispecs side-by-side.
- PageTabs (xaml + xaml.cs): This example is similar to the PageMultiSessions example showing how to utilise the Silverlight Tab control to display ispecs from multiple sessions in Tabs.

### 5.1.1 Events

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

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

CancelViewRendering property on the event argument causes the view to close and no further views will be shown until re-activated by the hosting application by calling the DisplayIspec method.

- StatusLine – Occurs after a response to a transaction has been received from the host system. This provides the opportunity for the application to check the status of the transaction and to customize the error handling. Setting the BypassStatusHandling property on the event arguments indicates the application is providing its own error handling and the default error handling provided by the View Controller will be bypassed.
- 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 browser 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 allows the application to specify an alternative form/view to load instead of the generated form/view. This allows designers to use tools like the Microsoft Expression Blend for creating alternative forms that bind to the data properties of the generated View Models as mentioned above, and still take full advantage of the capabilities of the CTC Silverlight View Controller and the CTC solution. Setting properties such as ViewName and ViewType on the event argument specifies the name and type of the alternate view to load.
- IspecReturnedByHost – Occurs when the host system returns an ispec that is different to the current ispec as a response to a transmit/transaction. This would occur when the host ispec does a recall of another ispec.
- HostSessionClosing – Occurs when the user closes the session by either closing the browser (IE only) or using the menu option CloseSession and before the session with the host system is closed. This gives the application the opportunity to take specific action before the session with the host system is closed.
- 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.
- UnsolicitedMessages - Occurs when unsolicited messages have been received from the host system. This provides the opportunity for the application to display the messages alerting the user.

### 5.1.2 Methods

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

- AutoConnect – Connects to the host system and establishes a session using run-time parameters specified in web.config.
- DisplayIspec – Displays the current ispec. This method is the default method for displaying ispecs that are returned by the host system in response to a transaction.
- OpenIspec – Opens or re-opens the specified ispec and returns an instance of the IspecView User Control. The caller displays the IspecView in a ContentControl on the user interface. This method is used for opening multiple ispecs within the same session and should only be used with ‘stateless’ EAE and AB Suite applications.
- CloseIspec – Closes and removes an ispec from the collection of open ispecs.

- CloseAllIspecs – Closes all ispecs and clears the collection of open ispecs.
- CloseSession – Closes the current session with the host system.
- GetUserControlInstance – Returns an instance of a UserControl that is part of the XAP package of an already open IspecView.
- Transmit – Transmits the current IspecView to the host system. This should only be used when there is a need to transmit an IspecView outside of the normal behaviour. The normal behaviour is to automatically transmit an IspecView to the host when the user clicks on a button or hits the Enter key.
- SetCurrentIspec – Sets the specified ispec name as the current Ispec. This should only be used when there is a need to set a different ispec as the current outside of the normal behaviour.

### 5.1.3 Properties

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

- CurrentIspecInfo – This provides information about the current active ispec such as the IspecModel, IspecView and IspecViewModel.
- CurrentIspecView – This points to the current active IspecView User Control.
- CurrentIspecViewContainer – This points to the ContentControl in which the current IspecView is displayed.
- DefaultIspecViewContainer – This can be set as the default ContentControl on the User Interface application in which IspecView controls can be displayed.
- OpenIspecs – This holds a collection of current open ispecs.

## 5.2 Multiple Ispecs

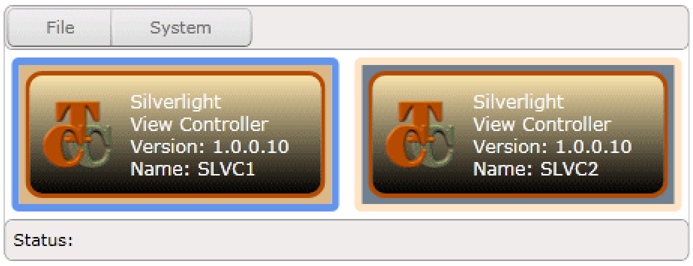
The CTC Silverlight View Controller allows the User Interface to open and interact with multiple ispecs. Depending on whether the host applications are designed and implemented as 'state full' or 'stateless' systems, the CTC Silverlight View Controller provides two ways of achieving this.

EAE and AB Suite systems allow a user to open and interact with only one ispec at a time within the same session. Therefore, to enable end users to open and interact with multiple ispecs, with 'state full' applications (i.e. the application maintains the state of the ispecs on the host using Global Work) the User Interface must allow for opening multiple sessions to the application. I.e. if the user requires two ispecs open at the same time, two sessions are required. See "Multiple Sessions" below.

With 'stateless' applications, the state of open ispecs can be maintained on the client side allowing the User Interface to keep multiple ispecs open within one session. See "Multiple Concurrent Open Ispecs in the Same Session" below.

### 5.2.1 Multiple Sessions

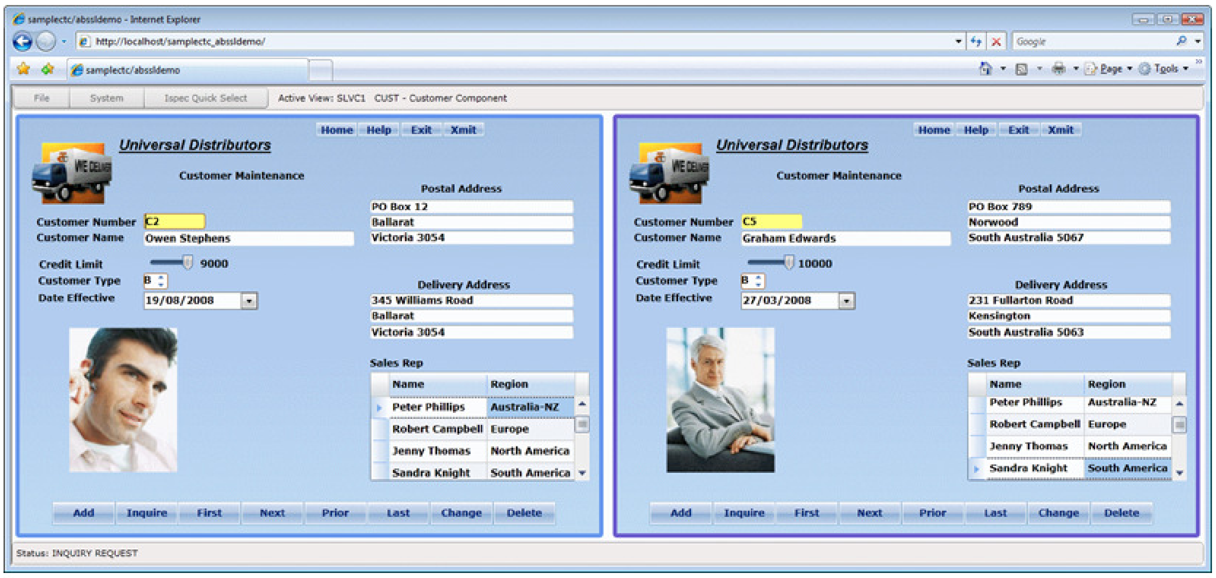
The CTC Silverlight View Controller is designed to run multiple sessions within the same User Interface application. For each session that is required, a CTC Silverlight View Controller is added to the User Interface application. As illustrated by the examples below, multiple View Controllers can be set up side-by-side or in tabs.



Silverlight
View Controller
Version: 1.0.0.10
Name: SLVC1

Silverlight
View Controller
Version: 1.0.0.10
Name: SLVC2

Multiple View Controllers side-by-side (Design View)



Active View: SLVC1 CUST - Customer Component

| | |
|---|---|
| Customer Number | C2 |
| Customer Name | Owen Stephens |
| Credit Limit | 9000 |
| Customer Type | B |
| Date Effective | 19/08/2008 |

Postal Address
PO Box 12
Ballarat
Victoria 3054

Delivery Address
345 Williams Road
Ballarat
Victoria 3054

Sales Rep

| Name | Region |
|---|---|
| Peter Phillips | Australia-NZ |
| Robert Campbell | Europe |
| Jenny Thomas | North America |
| Sandra Knight | South America |

| | |
|---|---|
| Customer Number | C5 |
| Customer Name | Graham Edwards |
| Credit Limit | 10000 |
| Customer Type | B |
| Date Effective | 27/03/2008 |

Postal Address
PO Box 789
Norwood
South Australia 5067

Delivery Address
231 Fullarton Road
Kensington
South Australia 5063

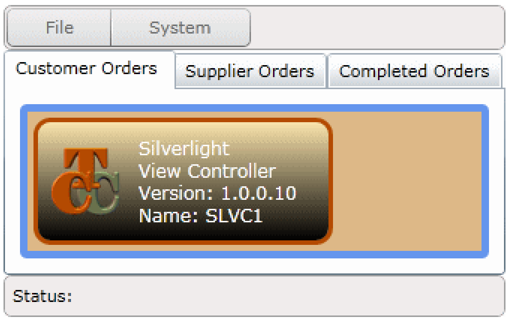
Sales Rep

| Name | Region |
|---|---|
| Peter Phillips | Australia-NZ |
| Robert Campbell | Europe |
| Jenny Thomas | North America |
| Sandra Knight | South America |

Status: INQUIRY REQUEST

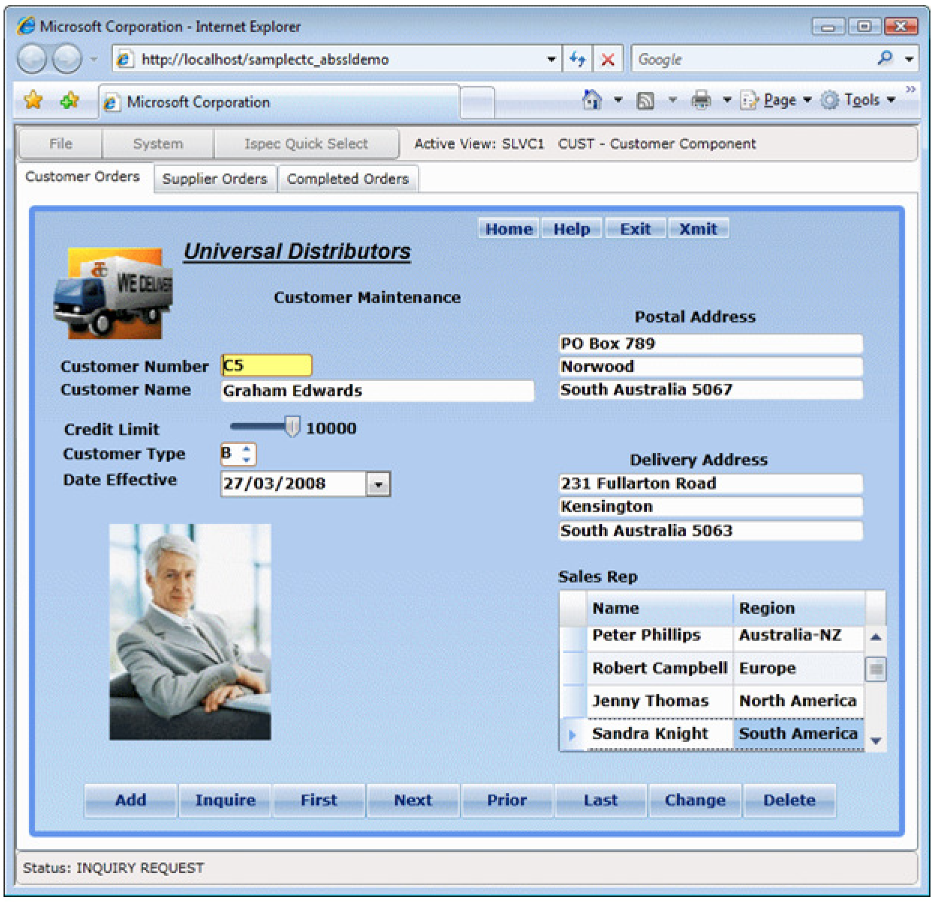
Multiple View Controllers side-by-side (Runtime View)

The example above shows a User Interface application with two View Controllers side-by-side allowing the end user to display two ispecs, one in each View Controller.



Silverlight
View Controller
Version: 1.0.0.10
Name: SLVC1

Multiple View Controllers using Tabs (Design View)



Active View: SLVC1 CUST - Customer Component

**Universal Distributors**

Customer Maintenance

| | |
|---|---|
| Customer Number | C5 |
| Customer Name | Graham Edwards |
| Credit Limit | 10000 |
| Customer Type | B |
| Date Effective | 27/03/2008 |

Postal Address
PO Box 789
Norwood
South Australia 5067

Delivery Address
231 Fullarton Road
Kensington
South Australia 5063

Sales Rep

| Name | Region |
|---|---|
| Peter Phillips | Australia-NZ |
| Robert Campbell | Europe |
| Jenny Thomas | North America |
| Sandra Knight | South America |

Status: INQUIRY REQUEST

Multiple View Controllers using Tabs (Runtime View)

The example above shows a User Interface application with three View Controllers in tabs allowing the end user to display three ispecs, one in each View Controller.

Based on the ‘standard’ Sample system, delivered with the installation of the CTC Silverlight Generator, are two examples of Silverlight User Interface applications. The examples are installed in the following folder:

“[ceroot]\CTC-Software\CTC Silverlight Client Generator\views\SilverlightApplication”

- PageMultiSessions – This shows multiple View Controllers side-by-side.
- PageTabs – This shows multiple View Controllers in tabs.

These examples can be used as the basis for customization to suit local requirements.

### 5.2.2 Multiple Concurrent Open Ispecs in the Same Session

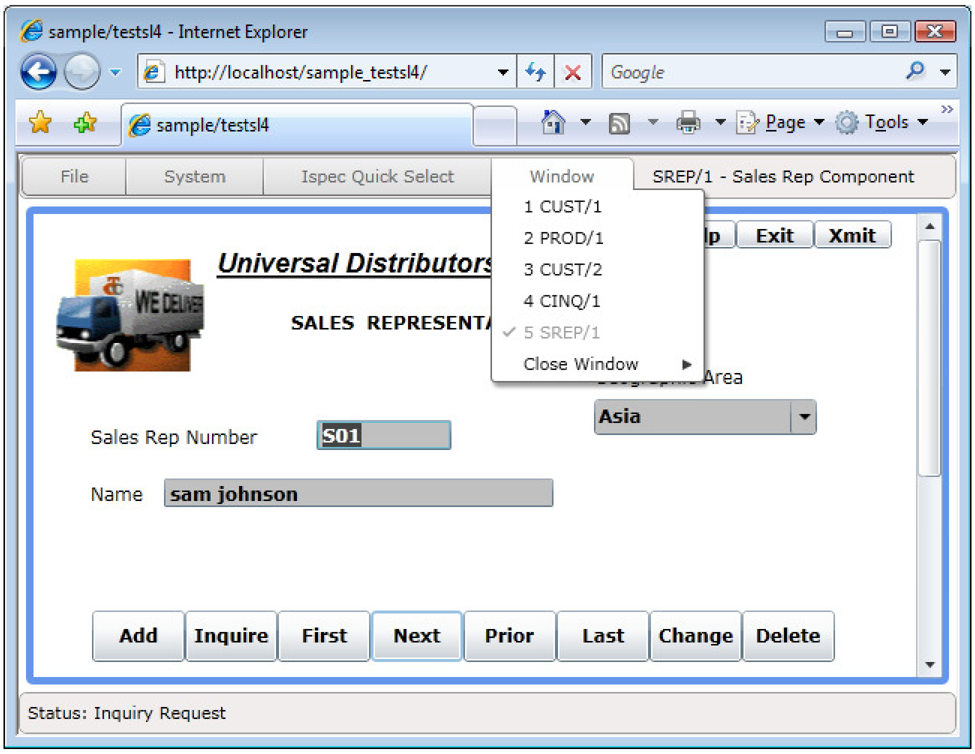
The CTC Silverlight View Controller is designed to manage multiple open ispecs at the same time within the same session. This can be utilized by EAE and AB Suite applications that are ‘stateless’ and therefore allowing the state of open ispecs to be managed on the client side within the User Interface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

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

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

#### 5.2.2.1 Ispecs Displayed in the Same Child-Window

Below is an example that shows 5 ispecs open at the same time. When the end user opens a new ispec, it is added to the 'Window' sub-menu. When the user selects an ispec from the 'Window' sub-menu, the ispec is made the current active ispec and is displayed.



Sales Rep Number S01

Name sam johnson

Asia

Status: Inquiry Request

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

"[ceroot]\CTC-Software\CTC Silverlight Client Generator\views\SilverlightApplication"

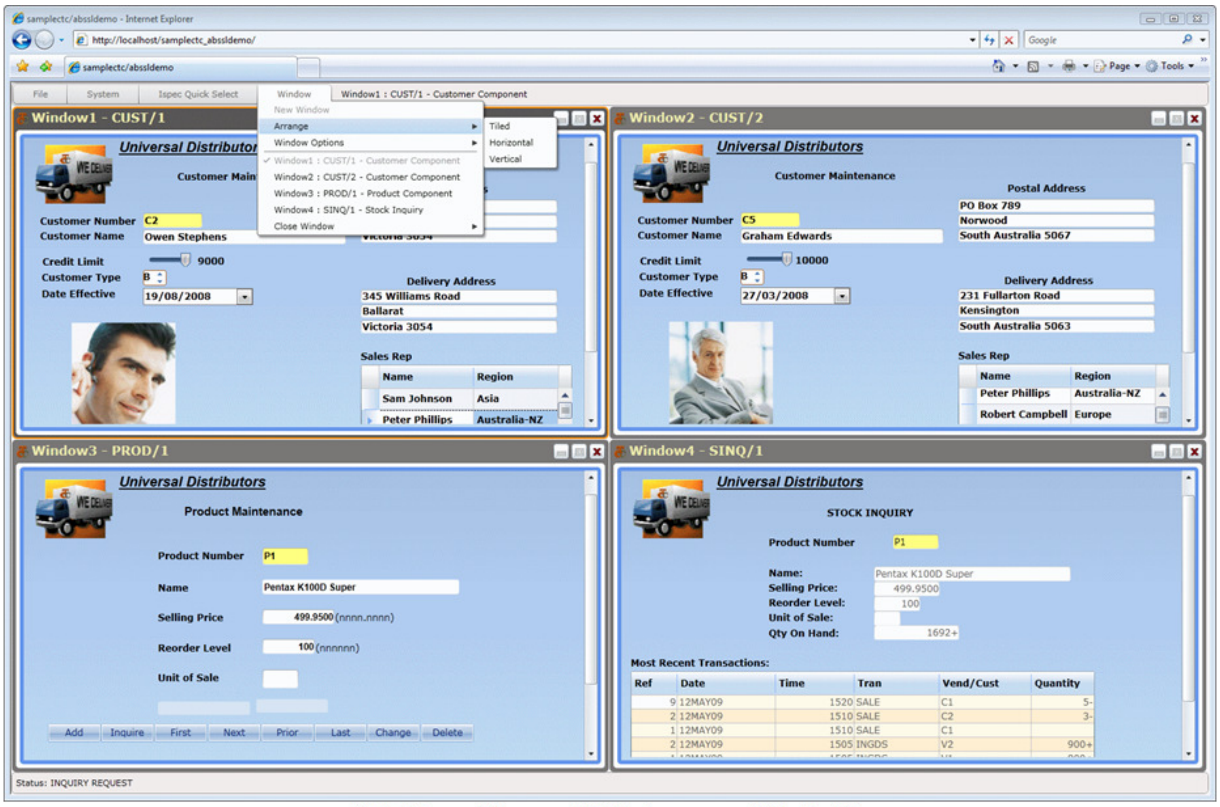
- PageMultiIspecs – This shows multiple open ispecs within the same session displayed in the same child-window.

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

#### 5.2.2.2 Ispecs Displayed in Separate Child-Windows

Below is an example that shows 4 ispecs open at the same time being displayed in separate child-windows. When the end user opens a new ispec, a new child-window can be opened and the ispec is shown within the window. The ispec window is added to the 'Window' sub-menu. When the user selects an ispec from the 'Window' sub-menu, the ispec is made the current active ispec allowing the user to transact with the ispec.

The user can open any number of child-windows. Using separate child-windows provides the opportunity for the end-user to arrange the windows tiled (side-by-side), horizontal or vertical as required.



Window1 - CUST/1 — Customer Maintenance

| | |
|---|---|
| Customer Number | C2 |
| Customer Name | Owen Stephens |
| Credit Limit | 9000 |
| Customer Type | B |
| Date Effective | 19/08/2008 |

Delivery Address
345 Williams Road
Ballarat
Victoria 3054

Sales Rep

| Name | Region |
|---|---|
| Sam Johnson | Asia |
| Peter Phillips | Australia-NZ |

Window2 - CUST/2 — Customer Maintenance

| | |
|---|---|
| Customer Number | C5 |
| Customer Name | Graham Edwards |
| Credit Limit | 10000 |
| Customer Type | B |
| Date Effective | 27/03/2008 |

Postal Address
PO Box 789
Norwood
South Australia 5067

Delivery Address
231 Fullarton Road
Kensington
South Australia 5063

Sales Rep

| Name | Region |
|---|---|
| Peter Phillips | Australia-NZ |
| Robert Campbell | Europe |

Window3 - PROD/1 — Product Maintenance

| | |
|---|---|
| Product Number | P1 |
| Name | Pentax K100D Super |
| Selling Price | 499.9500 (nnnn.nnnn) |
| Reorder Level | 100 (nnnnnn) |
| Unit of Sale | |

Status: INQUIRY REQUEST

Window4 - SINQ/1 — STOCK INQUIRY

| | |
|---|---|
| Product Number | P1 |
| Name: | Pentax K100D Super |
| Selling Price: | 499.9500 |
| Reorder Level: | 100 |
| Unit of Sale: | |
| Qty On Hand: | 1692+ |

Most Recent Transactions:

| Ref | Date | Time | Tran | Vend/Cust | Quantity |
|---|---|---|---|---|---|
| 9 | 12MAY09 | 1520 | SALE | C1 | 5- |
| 2 | 12MAY09 | 1510 | SALE | C2 | 3- |
| 1 | 12MAY09 | 1510 | SALE | C1 | |
| 2 | 12MAY09 | 1505 | INGDS | V2 | 900+ |

Multiple Ispecs in Separate Child-Windows arranged Side-By-Side

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

"[ceroot]\CTC-Software\CTC Silverlight Client Generator\views\SilverlightApplication"

- PageMultiIspecsChildWindows – This shows multiple open ispecs within the same session displayed in separate child-windows.

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

## 6 Deployment

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

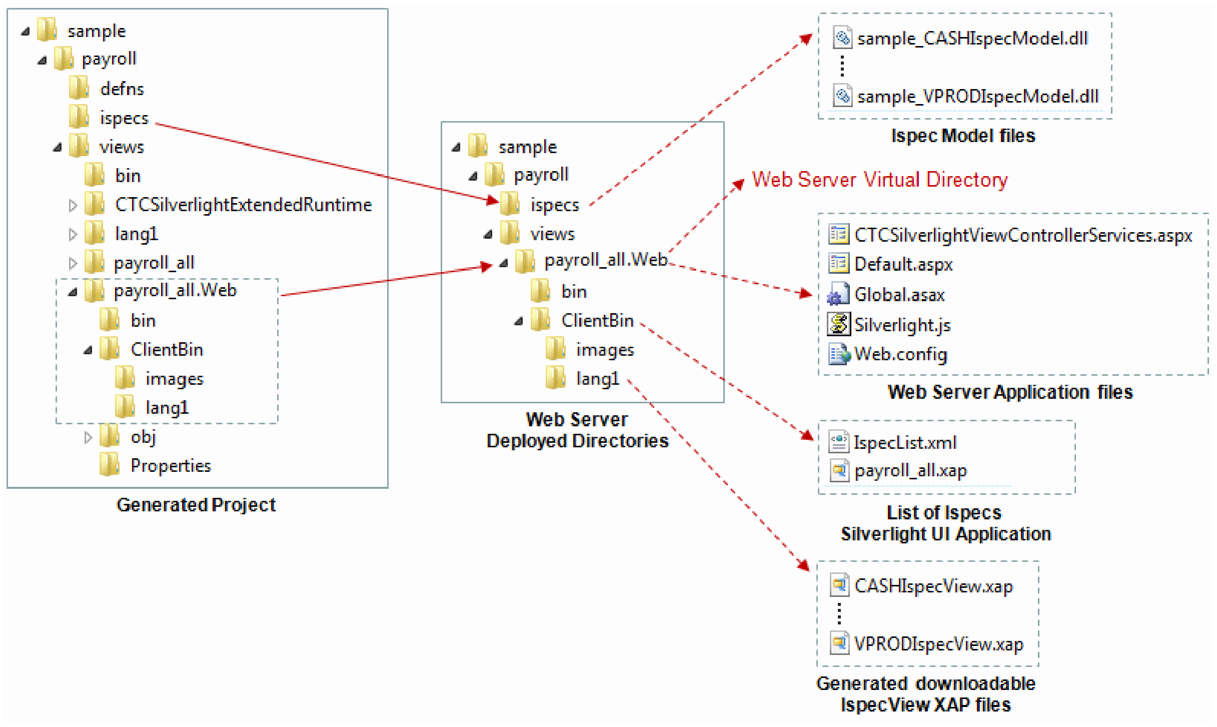
- Single Application
- Switch.To Application

## 6.1 Single Application

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

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

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



## 6.2 Switch.To Application

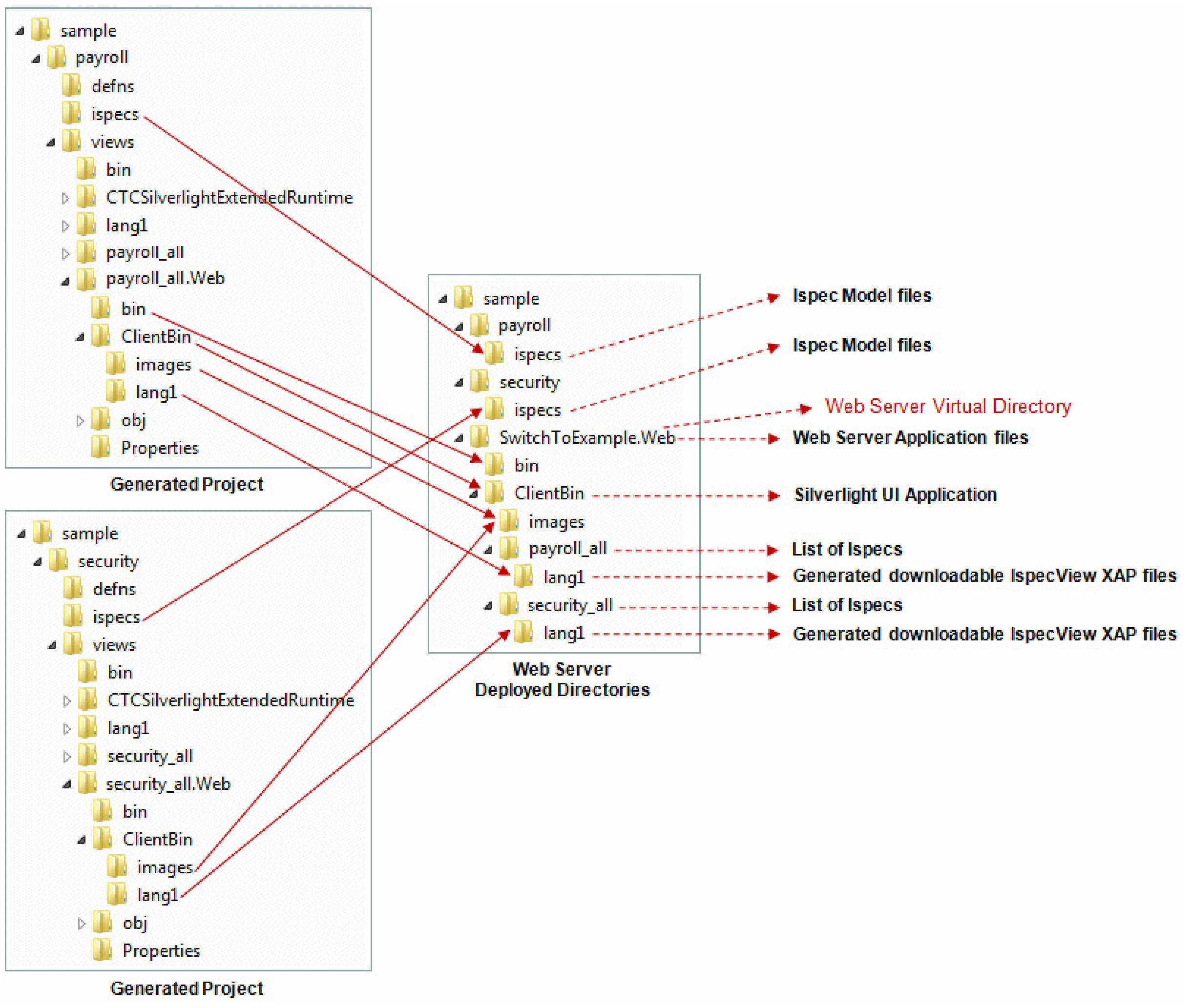
A Switch.To application is an application that consists of two or more generated application bundles, which the host system can switch between. As a Silverlight UI application is deployed from a web server, all files required for the application must be available within the same Virtual Directory. This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

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

- Ispec model files must be copied to the web server maintaining the same directory structure and folder names.

- Create a directory on the web server and configure this as the application Virtual Directory in IIS.
- Web Server Application files from the *.Web folder must be copied to the Virtual Directory as shown in the diagram below.
- Files in the bin folder must be copied to the Virtual Directory as shown in the diagram below.
- Silverlight UI Application files in the ClientBin folder must be copied to the Virtual Directory as shown in the diagram below.
- Files in the images folder from each of the generated projects must be copied to the Virtual Directory as shown in the diagram below.
- Create a directory for each of the generated projects under the Virtual Directory as shown in the diagram below. The directories must be named [application name]_[bundle name] (e.g. payroll_all and security_all).
- The IspecList.xaml file from the ClientBin folder in each of the generated projects must be copied to the Virtual Directory as shown in the diagram below.
- The generated downloadable IspecView XAP files from the language folders from each of the generated projects must be copied to the Virtual Directory as shown in the diagram below.

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



## 7 Custom Controls

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

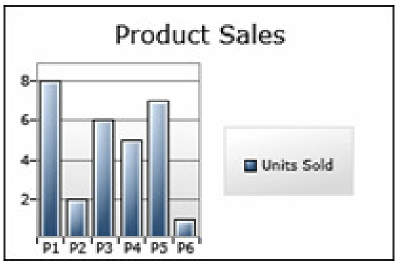
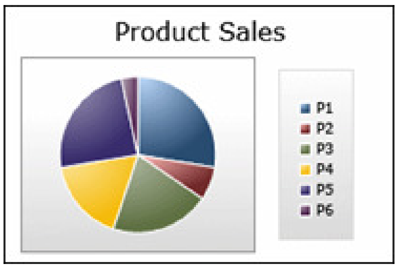
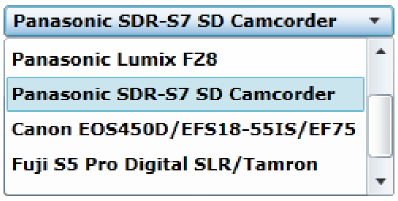
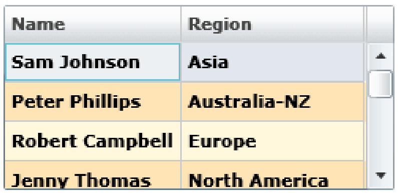
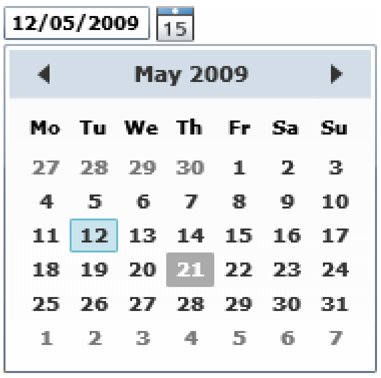
Custom Controls are used for substituting Standard Controls on the generated form when the User Interface requirements demand an interface that cannot be satisfied by the standard controls. Using the CTC Configurator, Standard Controls can be substituted with Custom Control (see the **CTC Silverlight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

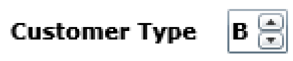
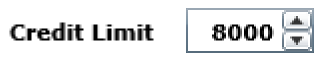
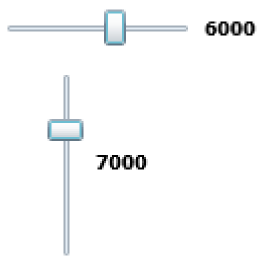
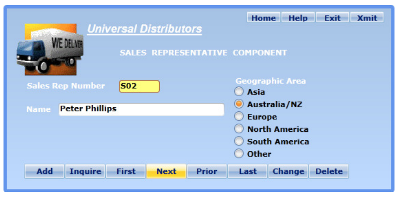
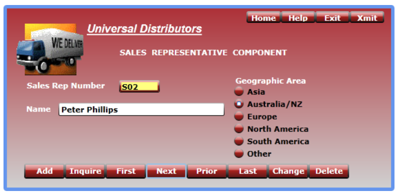
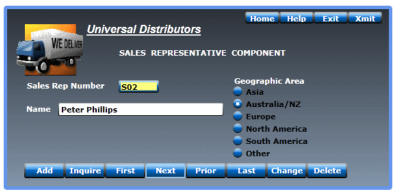
Included with the CTC Silverlight Client Generator are a number of Custom Controls. These can be used out of the box or changed to suit local requirement.

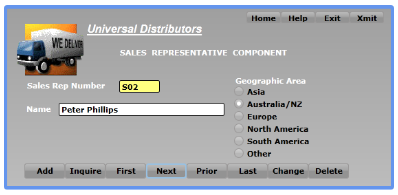
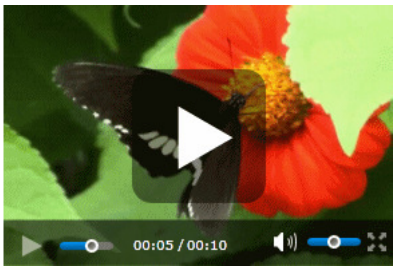
### 7.1 System Provided Custom Controls

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

Included in the list of custom controls are a number of controls from the Microsoft Silverlight Toolkit. The Silverlight Toolkit is a free download. Currently five controls from the toolkit that 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 Silverlight Toolkit see http://Silverlight.codeplex.com/.

| Control | Description |
|---|---|
| Charting   | This extends the standard ListBox control.<br><br>The Charting is a Silverlight Toolkit control. It provides easy charting capabilities of various chart types such as Area, Bar, Bubble, Column, Line, Pie and Scatter. |
| ComboBox Silverlight  Panasonic SDR-S7 SD Camcorder<br>Panasonic Lumix FZ8<br>Panasonic SDR-S7 SD Camcorder<br>Canon EOS450D/EFS18-55IS/EF75<br>Fuji S5 Pro Digital SLR/Tamron | This extends the standard CTC ComboBox control.<br><br>It allows the use of the ComboBox control provided by Silverlight as an alternative to the standard CTC ComboBox control.<br><br>The Silverlight ComboBox control has limited capabilities compared to the CTC ComboBox control. The Silverlight ComboBox does not allow the user to edit and type into the control and it doesn't allow the list to be shown as a fixed visible list. |
| DataGrid  Name / Region<br>Sam Johnson / Asia<br>Peter Phillips / Australia-NZ<br>Robert Campbell / Europe<br>Jenny Thomas / North America | This extends the standard ListBox control.<br><br>The Data Grid control is a flexible control with many options such as column headers, column sorting, column resizing, column reordering and styling. |
| DatePicker  12/05/2009  15<br>May 2009<br>Mo Tu We Th Fr Sa Su<br>27 28 29 30 1 2 3<br>4 5 6 7 8 9 10<br>11 12 13 14 15 16 17<br>18 19 20 21 22 23 24<br>25 26 27 28 29 30 31<br>1 2 3 4 5 6 7 | This extends the standard TextBox control where date input is required.<br><br>The DatePicker is a Silverlight Toolkit control. It provides a popup calendar control for easy date selection in date formats suitable for EAE and AB Suite. |

| | |
|---|---|
| DomainUpDown<br><br>Customer Type B | This extends the standard TextBox control.<br>The DomainUpDown is a Silverlight Toolkit control.<br>DomainUpDown adds 'Up' and 'Down' buttons to a TextBox that allows the user to select from a predetermined list of values. |
| NumericUpDown<br><br>Credit Limit 8000 | This extends the standard TextBox control.<br>The NumericUpDown is a Silverlight Toolkit control.<br>NumericUpDown adds 'Up' and 'Down' buttons to a TextBox that allows the user to select from a predetermined range of numeric values in predetermined increments. |
| HyperlinkButton<br>Client Tools Consultancy | This extends the standard Label control.<br>The HyperlinkButton is a Silverlight Toolkit control.<br>The HyperlinkButton custom control is used for substituting label controls where the label text has been specified as the html anchor control. I.e.: `<A HREF ='http://www.ClientTools.com.au ' >Client Tools Consultancy</A>` |
| Slider<br><br>6000<br>7000 | This extends the standard TextBox control.<br>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. |
| Themes Form<br><br>Universal Distributors<br>SALES REPRESENTATIVE COMPONENT<br>Sales Rep Number S02<br>Name Peter Phillips<br>Geographic Area: Asia / [✓ Australia/NZ] / Europe / North America / South America / Other<br> Home Help Exit Xmit<br>Add Inquire First Next Prior Last Change Delete<br><br>Universal Distributors<br>SALES REPRESENTATIVE COMPONENT<br>Sales Rep Number S02<br>Name Peter Phillips<br>Geographic Area: Asia / Australia/NZ / Europe / North America / South America / Other<br> Home Help Exit Xmit<br>Add Inquire First Next Prior Last Change Delete<br><br>Universal Distributors<br>SALES REPRESENTATIVE COMPONENT<br>Sales Rep Number S02<br>Name Peter Phillips<br>Geographic Area: Asia / Australia/NZ / Europe / North America / South America / Other<br> Home Help Exit Xmit<br>Add Inquire First Next Prior Last Change Delete | This extends the standard Form control.<br>The ThemesForm control is a Silverlight Toolkit control.<br>It allows the user to take advantage of the themes provided in the Silverlight toolkit and to easily apply different styles to the forms without making any changes to forms in the painter.<br>The following styles are currently available in the Silverlight toolkit:<br>• Bubble Crème<br>• Bureau Black<br>• Bureau Blue<br>• Expression Dark<br>• Expression Light<br>• Rainier Orange<br>• Rainier Purple<br>• Shiny Blue<br>• Shiny Red<br>• Twilight Blue<br>• Whistler Blue |

| | |
|---|---|
|   Home<br>Help<br>Exit<br>Xmit<br>Add<br>Inquire<br>First<br>Next<br>Prior<br>Last<br>Change<br>Delete Universal Distributors<br>SALES REPRESENTATIVE COMPONENT<br>Sales Rep Number: S02<br>Name: Peter Phillips<br>Geographic Area: Asia [✓ Australia/NZ] Europe North America South America Other | The theme is selected by specifying the Themes file on the ThemesForm control using the CTC Configurator. |
| Video Player<br> | This extends the standard Image control.<br><br>The VideoPlayer control is a custom control based on the Silverlight Media Element control extended with the typical media controls such as Start/Stop, Forward/Backward and Audio Volume control.<br><br>The VideoPlayer control include the following properties:<br>• MediaSource: The Uri of the video to play (i.e. videos/Butterfly.wmv, where videos is a sub folder of the ClientBin directory)<br>• ThumbnailSource: The Uri of a thumbnail image of the video to show initially (i.e. images/VideoThumbNail.png, where images is a sub folder of the ClientBin directory). When not specified, no thumbnail image will be shown.<br>• AutoStart: Boolean value specifying whether to automatically start playing the video. Default is False.<br>• Muted: Boolean value specifying whether to mute the audio initially. Default is False.<br>• AutoHide: Boolean value specifying whether to automatically hide the media controls when the mouse moves out of the controls. Default is True.<br>• CanScrub: Boolean value specifying whether to allow positioning the video by clicking on the time slider. Default is True.<br>• EnableFullScreen: Boolean value specifying whether to allow full screen mode. Default is True.<br>• LargeSpinnerIcon: Enum value specifying the kind of spinner icon to show when the video is streaming. Settings are: Circle, ProgressBar, None. Default is Circle. |

## 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 Silverlight 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 Silverlight Client Configurator** documentation.

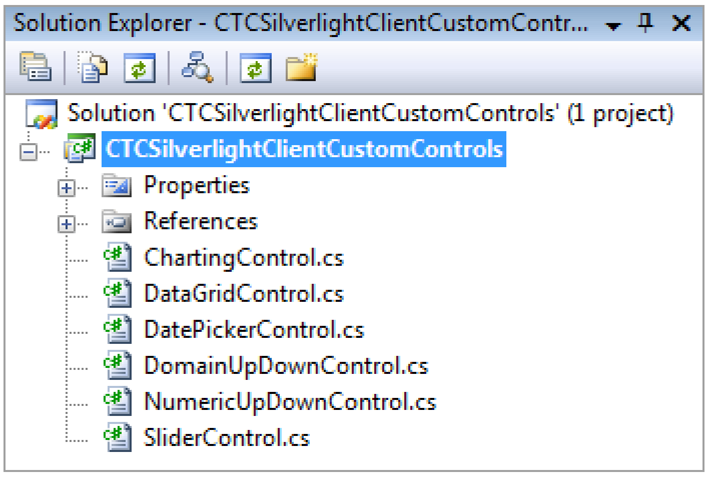
When creating Custom Controls, there are two distinct areas that need to be considered:

- The Generate Side
- The Runtime Side

### 7.2.1 Custom Controls – The Generate Side

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

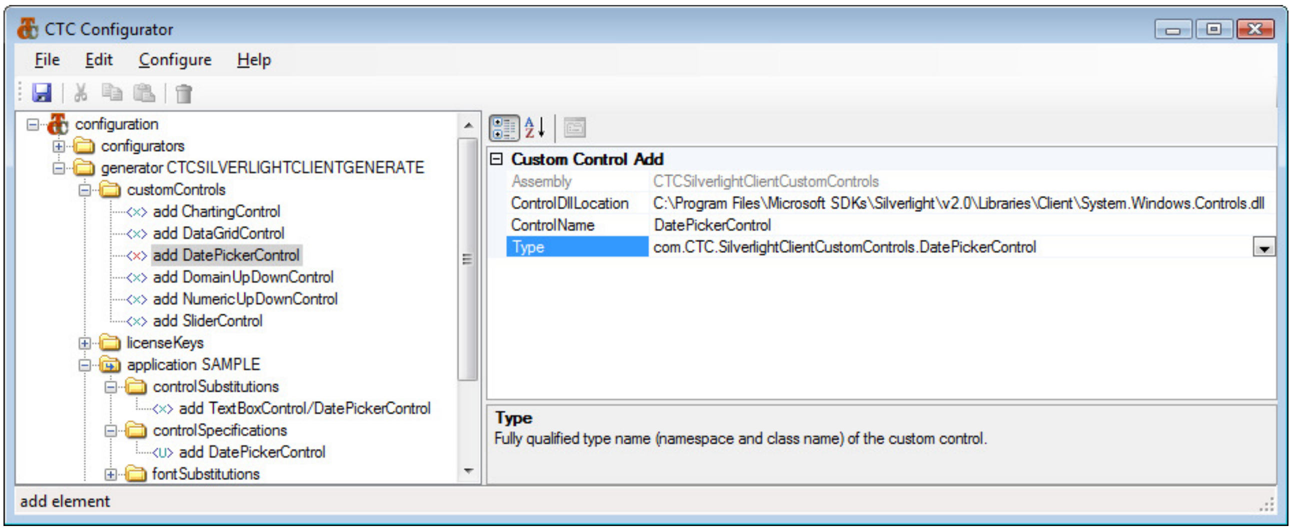

Solution Explorer - CTCSilverlightClientCustomContr...
Solution 'CTCSilverlightClientCustomControls' (1 project)
CTCSilverlightClientCustomControls
Properties
References
ChartingControl.cs
DataGridControl.cs
DatePickerControl.cs
DomainUpDownControl.cs
NumericUpDownControl.cs
SliderControl.cs

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



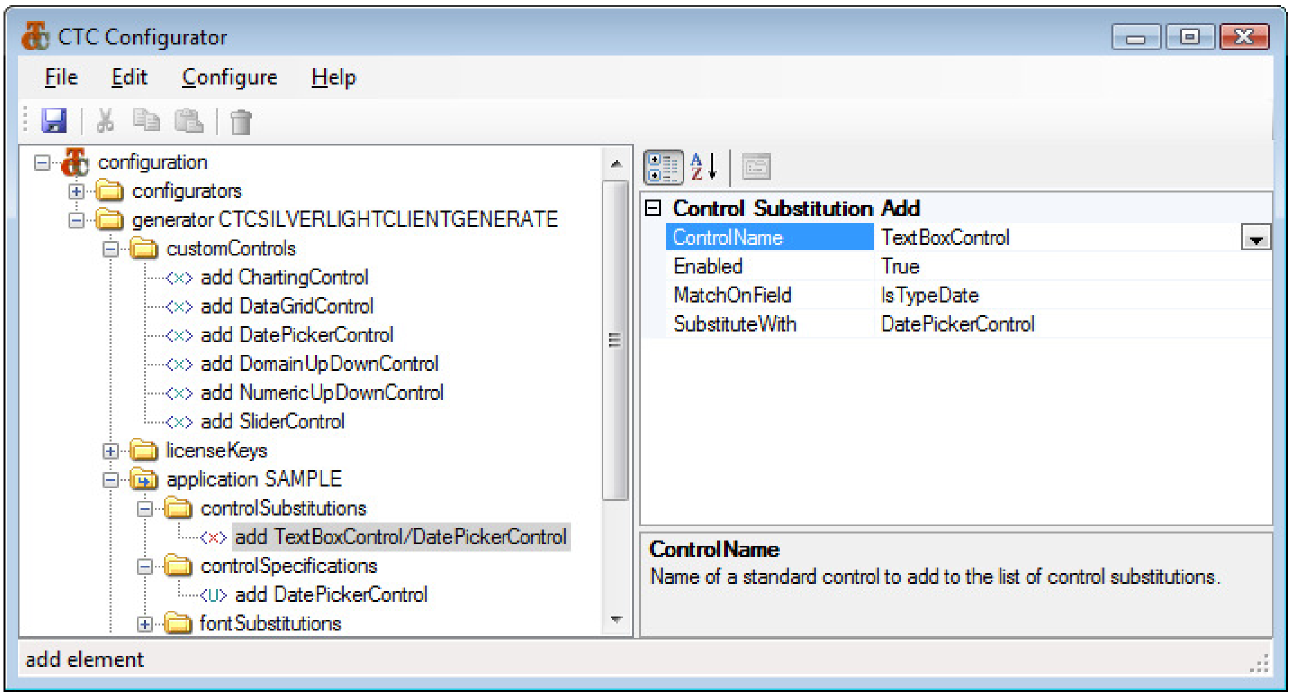
```
configuration
  configurators
  generator CTCSILVERLIGHTCLIENTGENERATE
    customControls
      add ChartingControl
      add DataGridControl
      add DatePickerControl
      add DomainUpDownControl
      add NumericUpDownControl
      add SliderControl
    licenseKeys
    application SAMPLE
      controlSubstitutions
        add TextBoxControl/DatePickerControl
      controlSpecifications
        add DatePickerControl
      fontSubstitutions
```

**Custom Control Add**

| | |
|---|---|
| Assembly | CTCSilverlightClientCustomControls |
| ControlDllLocation | C:\Program Files\Microsoft SDKs\Silverlight\v2.0\Libraries\Client\System.Windows.Controls.dll |
| ControlName | DatePickerControl |
| Type | com.CTC.SilverlightClientCustomControls.DatePickerControl |

**Type**
Fully qualified type name (namespace and class name) of the custom control.

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



```
configuration
  configurators
  generator CTCSILVERLIGHTCLIENTGENERATE
    customControls
      add ChartingControl
      add DataGridControl
      add DatePickerControl
      add DomainUpDownControl
      add NumericUpDownControl
      add SliderControl
    licenseKeys
    application SAMPLE
      controlSubstitutions
        add TextBoxControl/DatePickerControl
      controlSpecifications
        add DatePickerControl
      fontSubstitutions
```

**Control Substitution Add**

| | |
|---|---|
| ControlName | TextBoxControl |
| Enabled | True |
| MatchOnField | IsTypeDate |
| SubstituteWith | DatePickerControl |

**ControlName**
Name of a standard control to add to the list of control substitutions.

### 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.