SYSPRO 8

Application Designer

Overview Guide

Version: 005

Last Published: April 2023

SYSPRO™

SYSPRO Help and Reference

Copyright © 2023 SYSPRO Ltd

All rights reserved

No part of this document may be copied, photocopied, or reproduced in any form or by any means without permission in writing from SYSPRO Ltd. SYSPRO is a trademark of SYSPRO Ltd. All other trademarks, service marks, products or services are trademarks or registered trademarks of their respective holders.

SYSPRO Ltd reserves the right to alter the contents of this document without prior notice. While every effort is made to ensure that the contents of this document are correct, no liability whatsoever will be accepted for any errors or omissions.

This document is a copyright work and is protected by local copyright, civil and criminal law and international treaty. This document further contains secret, confidential and proprietary information belonging to SYSPRO Ltd. It is disclosed solely for the purposes of it being used in the context of the licensed use of the SYSPRO Ltd computer software products to which it relates. Such copyright works and information may not be published, disseminated, broadcast, copied or used for any other purpose. This document and all portions thereof included, but without limitation, copyright, trade secret and other intellectual property rights subsisting therein and relating thereto, are and shall at all times remain the sole property of SYSPRO Ltd.

WHAT'S CHANGED?

SYNOPSIS	READ MORE...
Adding custom applications to the Menu, in either the Desktop or Web UI.	Adding Custom Applications to the SYSPRO or Avanti Menu

ANNOTATION

The contents of this document are subject to change.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Contents

Introducing the Application Designer 5

Benefits and features 5
Understanding the Application Designer 5
The Basics 6
Designing for Avanti 8
Creating a New Application 9
Designing a Toolbar 18
Designing a Context Menu 24
Creating Task Dialogs 25
Creating Avanti Web Views 33
Using Data Sources 42
Adding Custom Applications to the Desktop or Web UI Menu 75
Getting Help 77

Programming the Application 78

Application Designer VBScript Events 79
How to use a Dialog Box 81
Event Cycle Processing 83
Some Basics 84

Some VBScript Suggestions 86

Functions and Classes 86
Comment the Code 86
AppDesigner_OnUserEvent 87
The Use of Constants 87
Re-useable VBScript Functions and Classes 88
Debugging and Logging Progress 89
Using Persisted Variables 90
Retrieving Setup Options 91

Understanding the Controls 93

Chart Control 93
List View Control 112
Form Control 124
VBScript Hints 137
Tree View Control 138
Syntax Editor Control 144
XAML Control 146
Document Viewer 164
Image Viewer 166
Tile Control 167
Flowgraph Control 168
WebBrowser Control 171

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Drop Files Control 172

Working with Projects 175

Managing Projects 175
Creating New Projects 176
Maintaining Projects 177

Sample Apps 180

Appendix A – VBScript code snippets 181

App Comment section 181
Catching EventType 181
Displaying OnUserEvent Parameters 181
Get Local Windows Temp Directory Name 182
Toolbar Constants 182
Writing Text to a File – Simple Function 182

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Introducing the Application Designer

The Application Designer is an application development environment that harnesses the SYSPRO User Interface, its business logic, its event handlers, and a Microsoft scripting language. It presents them in an easy-to-use interface which generates applications that run in SYSPRO.

It is designed to reduce the complexity of coding by providing a simple (low code) interface into all the SYSPRO functionality.

Benefits and features

- Improved productivity as it allows developers to create a new UI for a program in a matter of minutes rather than hours.
- No additional technology required to produce and run applications.
- Improved storyboarding:
  Because of the rapid time to create a UI, it can make it easy to produce a prototype and get sign-off approval for it.
- The complexity of program development is hugely reduced, as a single text file describes the entire application and can be easily deployed on a target machine.
- Multiple applications can be handled using Projects to deploy a complete solution to the end-user.
- Applications can be designed to run in both the SYSPRO Desktop and SYSPRO Web UI (Avanti).
- The Application Designer is not only used by partners and customers; it is also used by SYSPRO development to build standard SYSPRO applications. It is therefore, a robust, tried and tested platform for all types of applications.

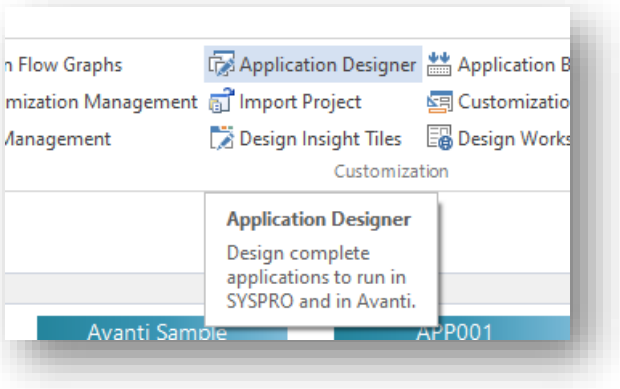
Understanding the Application Designer

The designer is used to create applications to be used by end-users who in turn might wish to further customize the application by adding custom form fields, adding their own customized panes and a myriad other ways of customizing and personalizing the application.

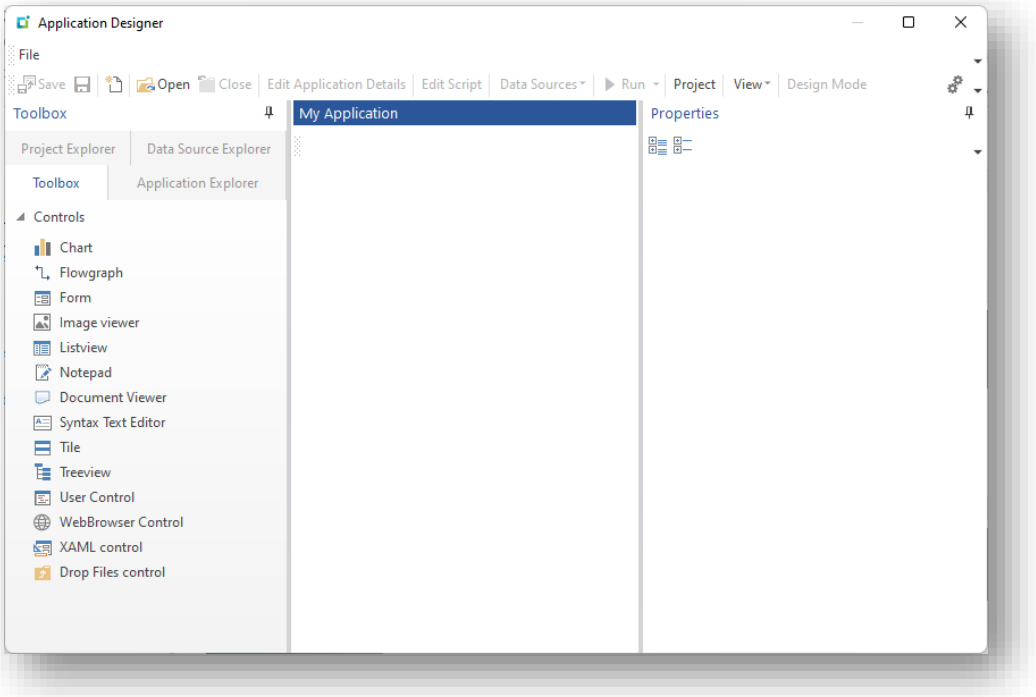
© SYSPRO. All Rights Reserved. All trademarks are recognized.

The Basics

To design an application, you call **IMPIDE** from SYSPRO's main menu, or click on **Application Designer** in the **Administration** tab of the Ribbon Bar.



The initial view of the Application Designer looks like this:



Toolbox

The **Toolbox** tab shows the list of controls that can be added to an application. Up to 30 controls can reside in a single application.

Controls are arranged in your application within a docking pane layout, meaning that each control is contained in a docking pane which in turn is contained in a layout.

You can arrange these docking panes in any way you like within the application area simply by clicking on the docking pane header and dragging it to a new position, or by grouping them

© SYSPRO. All Rights Reserved. All trademarks are recognized.

together in a TAB sequence. However you arrange your panes is exactly how it will look when you run the application.

If you run an application whose option **In Development** is switched off, then any saved docking layout will be applied as the application loads. Generally this all works okay, but if the application contains either new or deleted panes then it's possible that the application will not work correctly as the saved docking layout will not include these changed panes.

The Application Designer runtime will detect if the number of panes in the saved layout do not match that in the Application itself and will then automatically reset the docking layout.

Note: This automatic Reset Layout will not occur if there are any customized panes detected in the saved docking pane layout.

Application Explorer

The **Application Explorer** tab shows all the controls and toolbar buttons that have been added to an application. This is used to easily navigate to any control or toolbar button with a single click.

Project Explorer

The **Project Explorer** tab shows all the projects and their resources. Click on a Project to edit it or click on an application to open it for editing.

Data Source Explorer

The **Data Source Explorer** tab shows where a data source will be executed in the application.

Click on the hyperlinked execution point and this will navigate to the form field, the toolbar button or list view.

Click on the Data source name hyperlink to edit the data source.

My Application

The area marked as **My Application** is where the application being designed will reside.

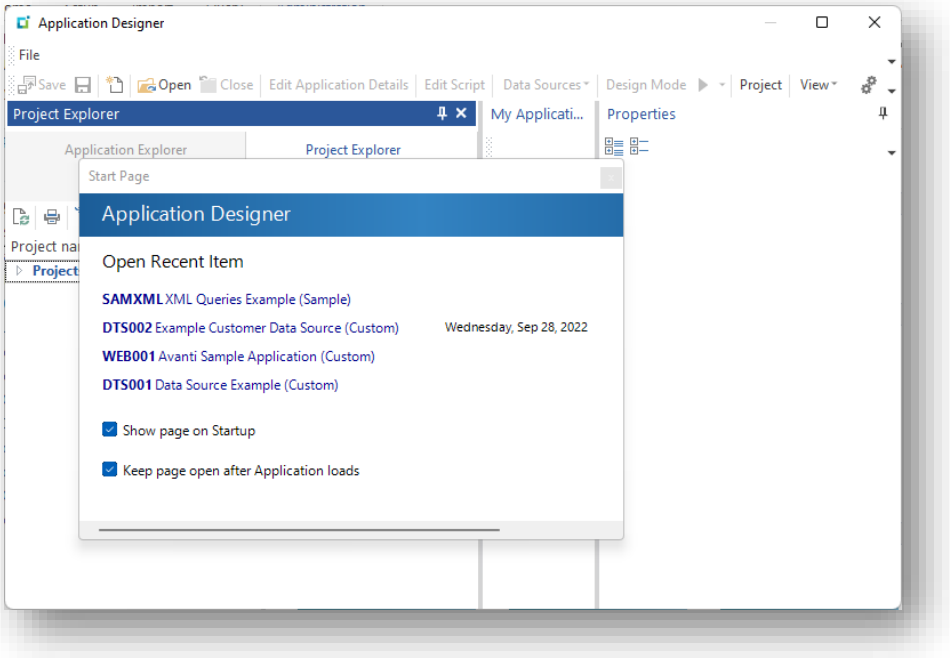
Properties

The **Properties** window will reflect the properties of the current control being edited or the main application details.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

START PAGE

The Start Page window shows your most recent applications that have been opened:



With a single click on any item, you can view any application and edit it.

Optionally, you can also define if the Start Page should automatically be displayed as the Application Designer loads; and if it should remain open after opening an application for maintenance.

DESIGNING FOR AVANTI

The Application Designer can only be run from within a SYSPRO Windows client. To create an Avanti Web View for an application, proceed as follows:

1. Create the application using the **Application Designer** from within SYSPRO.
2. Create Web Views for the primary layout and all modal windows of the application from within the Designer.
3. Optionally, select the option to embed the web views(s) in the application.

See the section Creating Avanti Web Views for more information.

Note: The following controls are currently not supported in an Avanti Web View:

- Flowgraph
- Image Viewer
- Document Viewer
- Tile Control
- Web Browser control
- XAML control
- Drop Files control

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Creating a New Application

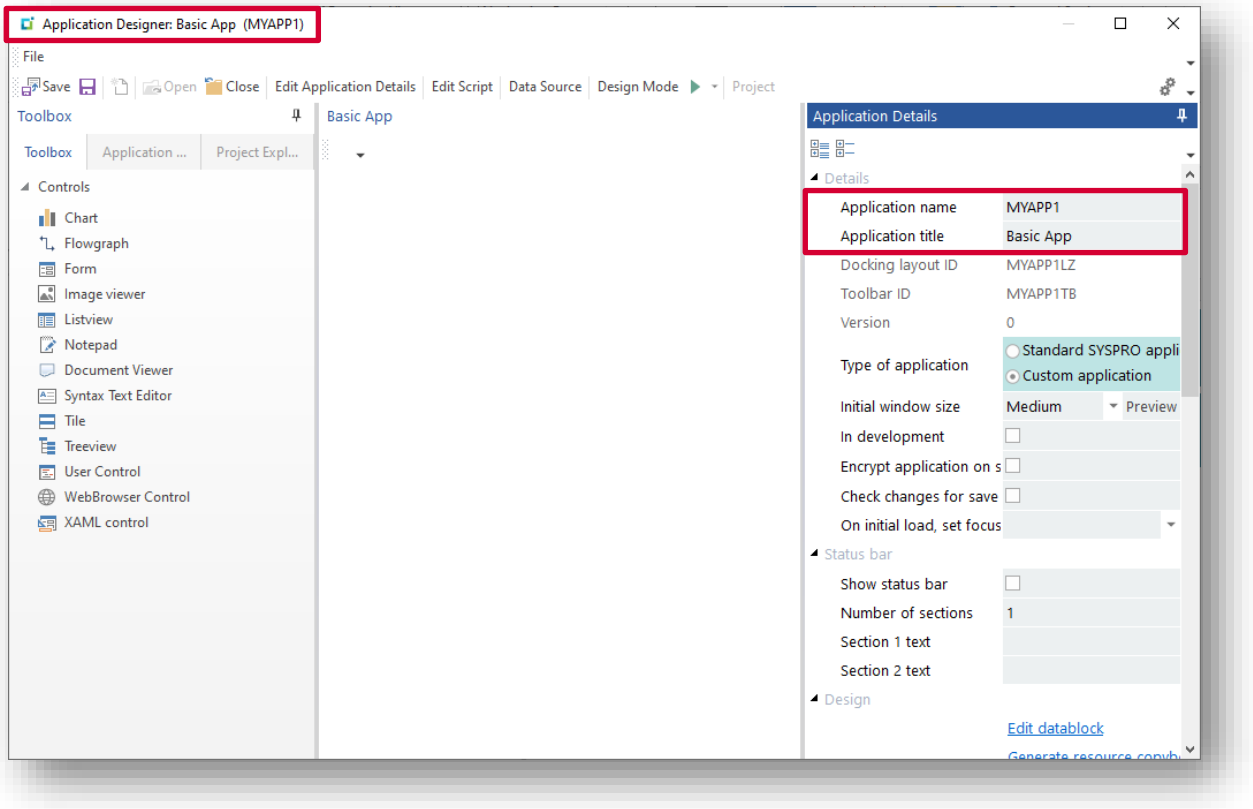
Click the **New** button to start creating a new application.

Enter your application name (the example is **MYAPP1**).

Considerations:

- This must be exactly 6 characters long and cannot contain special characters.
- This is the name of your application when you run it from a SYSPRO menu or in Avanti.
- Once the application has been saved, you cannot subsequently change the name.
- For a Custom application, the Application name must not begin with any of the standard SYSPRO module names (such as IMP, ARS, APS, INV etc.). This is to prevent any standard SYSPRO application from being overwritten.

Now enter the title of the application, which will appear as the window title:



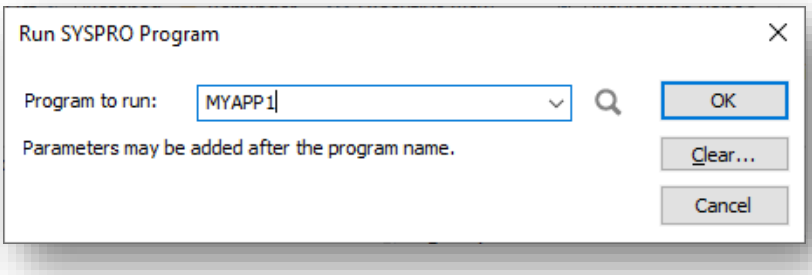
Note: It is important therefore that a naming convention is established so that if applications are being written by multiple developers, there is no clash in application names.
This is particularly important when using the project export and import function, as it would be very easy to overwrite an application written by someone else.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Testing the Application

There are two ways to test the application:

1. Select **Save** to save the application, and then exit out of the Application Designer. You can now run your application by pressing **CTRL+R** in the main SYSPRO menu and entering your program to run:



2. Alternatively, press **F5** to run the application.
   This will automatically save the application and then run it in a separate instance of SYSPRO, the advantage being that you don't leave the application designer.

Your application will look like this:

Click **X** to close the application.

You are now ready to start changing some detail about the application, as well as adding controls to the application.

Changing Application Details

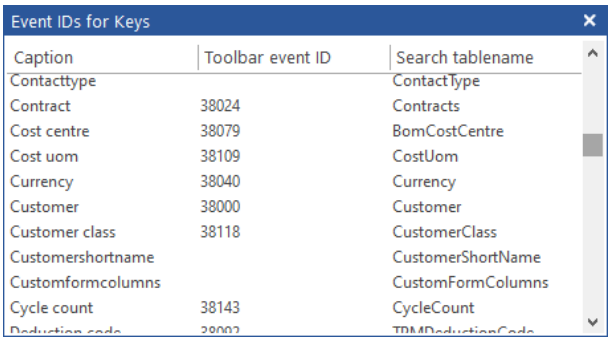
In the Application Designer, click on **Edit Application Details** to change the properties of the application.

The following provides details for each option available:

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Property	Explanation
DETAILS	
Application title	This indicates the title of the application that will appear in the window heading and be used to identify the application in the **Recent Programs** list in SYSPRO.
Docking Layout ID Toolbar ID	Each control and its associated (optional) toolbar have unique IDS. You don't generally need to know these IDs, but they can be referenced in script code, which is why they are generated by the designer, and you cannot change the values.
Version	This number increments every time the application is saved, unless the **In Development** option is enabled.
Type of application	There are two types of applications: ▪ **Standard** applications supplied by SYSPRO ▪ **Custom** applications **Note:** Only the Custom option will be available outside of SYSPRO's development department. Custom applications are saved as TEXT files in the `…base\settings\AppDesigner` folder on the local machine and then copied to the same-named folder on the server (if in client/server mode).
Initial window size	Select the appropriate initial window size for your application. Remember this is just the default view for the user; the user can resize and position the application window as required. Click the **Preview** button to see how the application will look – this floats the application window with the selected size. Double-click the docking pane caption to close the preview.
In development	Select this option to indicate that the application is 'in development'. When running the application and while this option is set, the application window position and any changes to Forms, List views, Toolbars and docking layouts will not be saved when the application exits. This is useful while developing the application. In addition, the Version number will not be incremented when the application is saved. **Note:** Remember to switch this option off before the application is deployed.
Encrypt application on save	This indicates that the application will be saved as both a text and an encrypted file (with an extension of `.snk`). The encrypted version has the advantage of being deployable to a site and run as a normal application, but the text cannot be humanly read or opened in the Application Designer. This protects the IP of the application.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Property	Explanation
Check changes for save	This provides an automatic SAVE CHANGES mechanism. When clicking on the NEXT or PREVIOUS navigation buttons in the toolbar (and if the contents of any list views, forms, notepads, or syntax editor have changed) then a 'Save Changes' task dialog will be shown. Event `ToolbarSaveChanges` is raised if the SAVE button is pressed Event `ToolbarCancelChanges` is raised if CANCEL button is pressed.
On initial load, set focus to	Select the control to set focus to after the application loads. For an editable Form, this means start editing on the first field on the form. For an editable List view, this means start editing on the first row in the list view. For a toolbar, this means setting focus to the edit control.
STATUS BAR	
Show status bar	Select this to show a status bar for the application.
Number of sections	Select one or two sections.
Section 1 text	Enter text for the first section. You can dynamically change this text using the script callout `ToolbarSetStatusBarText`. This script statement will show the text **Ready** in section **1** of the status bar: `call CallSYSPROFunction(ToolbarSetStatusBarText, "DTS001TB", "1|Ready")`
Section 2 text	Enter text for the second section. You can dynamically change this text using the script callout `ToolbarSetStatusBarText`.
DESIGN	
Design toolbar	Click to design the toolbar for the main application window. *See the Designing a toolbar section for more information.*
Event IDs for keys	Click to see the event IDs and caption names for all key fields. Key fields are used for Predictive Search and Smart Links. *For example: A caption of **Customer class** in a form or grid will enable predictive search for that field; similarly, a toolbar event ID of 38118 will enable predictive search in a toolbar.* 

Event IDs for Keys

Caption	Toolbar event ID	Search tablename
Contacttype		ContactType
Contract	38024	Contracts
Cost centre	38079	BomCostCentre
Cost uom	38109	CostUom
Currency	38040	Currency
Customer	38000	Customer
Customer class	38118	CustomerClass
Customershortname		CustomerShortName
Customformcolumns		CustomFormColumns
Cycle count	38143	CycleCount

© SYSPRO. All Rights Reserved. All trademarks are recognized.

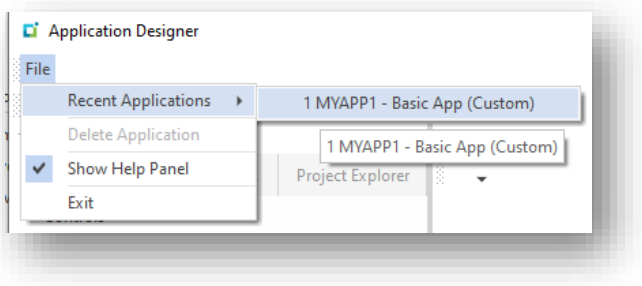
Property	Explanation
SCRIPTING	
Edit script	Click to edit the script associated with the application. *See the Programming the Application section for more information.*
Form OnGainFocus event required	This indicates that the script should be sent the `GainFocus` event for a form. If enabled: every time the user sets focus to any editable form field, then this event will be sent to the script. **Note:** This may impair performance of the application, so leave this option unchecked unless the application absolutely requires this event.
Toolbar OnGainFocus event required	This indicates that the script should be sent the `GainFocus` event for a toolbar entry field. If enabled: every time the user sets focus to any editable toolbar field, then this event will be sent to the script. **Note:** This may impair performance of the application, so leave this option unchecked unless the application absolutely requires this event.
AVANTI PROPERTIES	
Select Web View to design	All applications that run in Avanti have a 'Primary Layout' Web View (which is the main application's view) and a Web View for each of the Dialog Boxes defined in the application. From the dropdown list, select the Web View that you wish to design, then click **Design Web View**.
Design Web View	Use the Visual Designer to design the selected Web View. ▪ SYSPRO applications: The web view templates will be located in the `...base\SAMPLES` folder. ▪ Custom applications: The web view templates will be located in the `...base\settings\AppDesigner` folder. *See the Designing a Web View section for more information.*
Define Modal Windows	If you need more sophisticated modal windows in Avanti, see the Creating additional modal windows for Avanti section for more information.
Embed Web View in application	This indicates that the web views designed for this application are embedded within the application text file. This makes it easier to deploy the application. **Note:** If you do not select this option, Web Views designed for the application will need to be deployed separately.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Property	Explanation
Show as modal window in Avanti	This indicates that this application should be shown in a modal window when running as a Web View in Avanti. This is the appropriate choice if the application window is small enough not to have to consume the entire web page, as a modal window in Avanti means the Home Page workspace does not need to be refreshed.
APPLICATION PROPERTIES *These properties are shown in the **About this Application** window (selected from the Application's Gear icon).*	
Author	Enter the author of the application.
Release version	Enter the release version of the application.
Show notes in About window	This indicates that the **Notes** will be shown in the 'About this Application' window.
Notes	These notes are available for you to annotate your application. They do not affect the running of the application.

ADDING CONTROLS TO AN APPLICATION

In the Application Designer, click on **File > Recent Applications** and select **MYAPP1**.

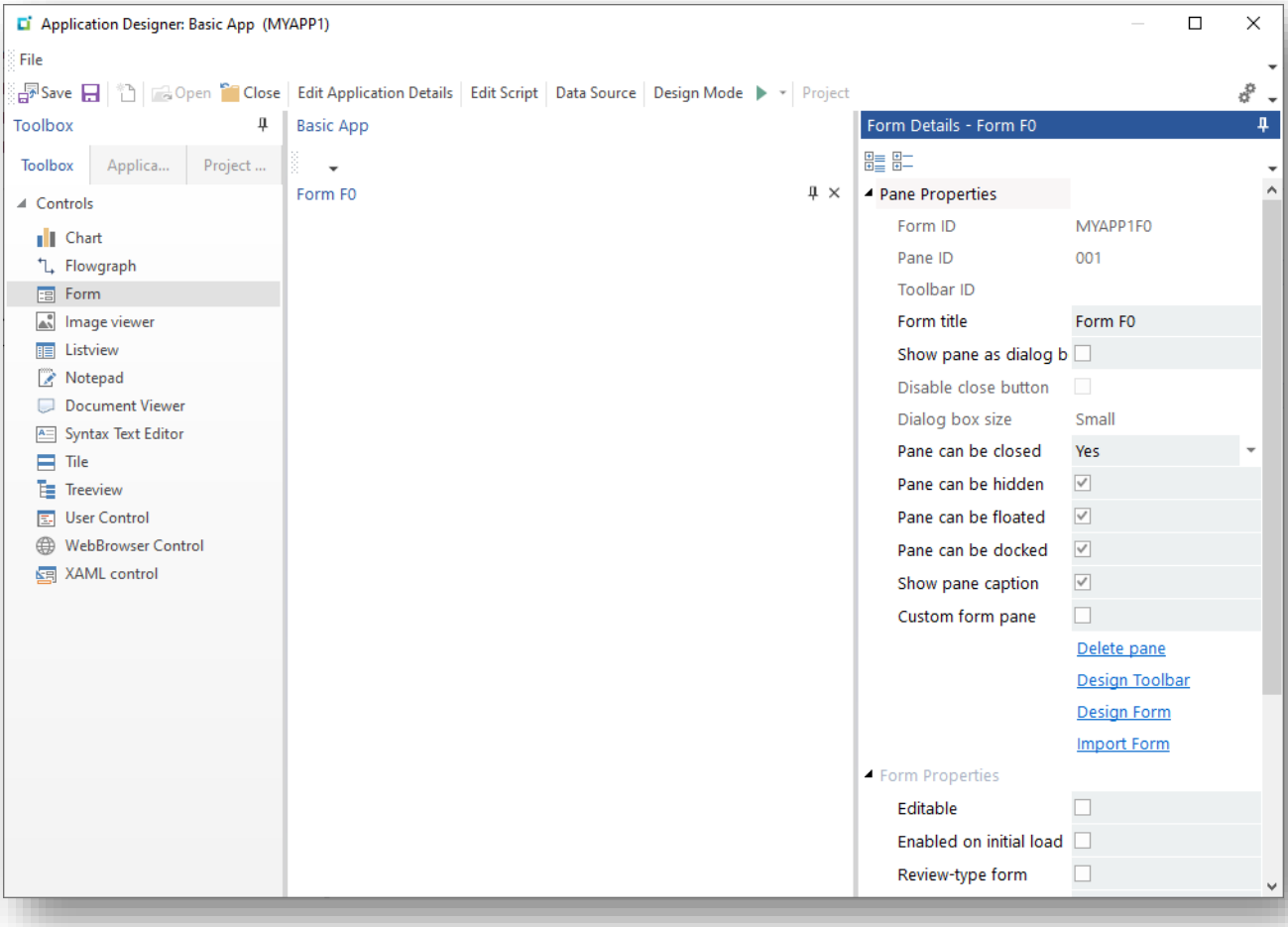


You will notice the option **Show Help Panel**. Just about every option in the designer has form field help. After a while, and when you've become familiar with the designer, you may want to switch off this option which will give you more space in the Properties forms.

To add a control to an application, click on the required Toolbar item. A control is always added to the top left of the application and from there you can position it anywhere as required. You can also make the pane 'closed' by default by clicking on the **X**, or 'hidden' by clicking on the **Pin** button. Remember that however you change a pane is how it will appear when a user first runs the application.

In the following example, a Form has been added to the application:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



All controls have a common set of Properties (i.e. Title, Pane properties, etc.) and then a set of specific properties.

The following explains the common properties:

Property	Explanation
Title	The title is shown in the docking pane header and will be automatically translated.
Show pane as dialog box	This indicates that the pane will only be shown when the application invokes the **`ShowDialogBox`** script callout. Regardless of the appearance of the pane when designing, it will be automatically closed and set as 'disabled' as the application loads. A dialog box is a temporary window to retrieve user input or show some information. *See the section How to use a Dialog Box for more information.*
Disable close button	This indicates that the CLOSE button (X) on the window is not shown, meaning that the dialog box can only be closed by script code. This is useful, for example, if you wish data to be checked before the dialog box is closed.

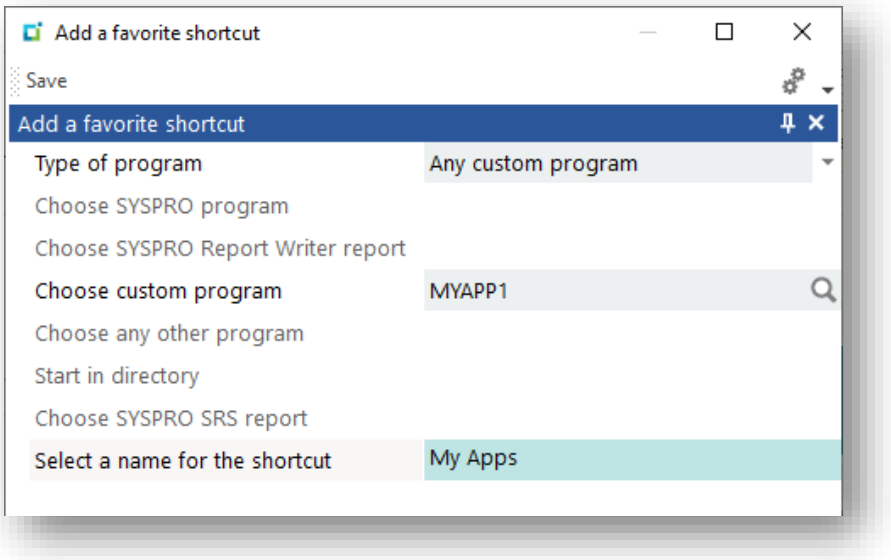
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Property	Explanation
Dialog box size	Select the initial size of the window, according to the requirements of the control to be hosted in the pane. This is just the initial size; the user can resize and reposition the window as required when running the application. *See the In development option for more information.*
Pane can be closed	This indicates if the pane can be closed by the user by clicking the **X** (the default option). Select **No** if you don't want the user to close this pane, perhaps if the pane content is central to the use of the application. Select **Always closable** if the pane can be closed, even if Roles are in use. It is recommended that at least 1 docking pane cannot be closed, so that the user cannot remove all panes and be left with a blank application.
Pane can be floated	This indicates the pane can be dragged and floated.
Pane can be docked	This indicates the pane can be docked to another position in the layout.
Show pane caption	This ensures the pane caption is to be shown. If unchecked, a user will not be able to reposition the docking pane.
Delete pane	Click this to delete the pane.
Design Toolbar	This lets you design a toolbar for the pane. *See the Designing a toolbar section for more info.*

Running the Application

You can add your application to a menu, or your Favourites Tile.

When adding a program select the **Custom program** option, as shown below:



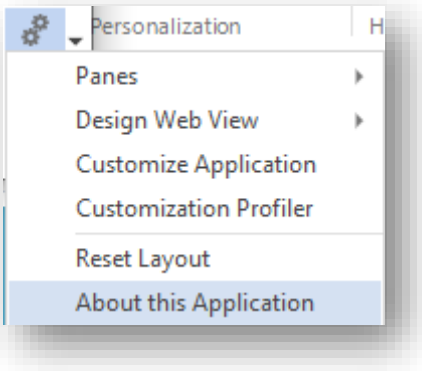
© SYSPRO. All Rights Reserved. All trademarks are recognized.

An application designed using the Application Designer works in a similar manner as any SYSPRO application, so much so that a user will not easily be able to distinguish between a standard SYSPRO and a custom application.

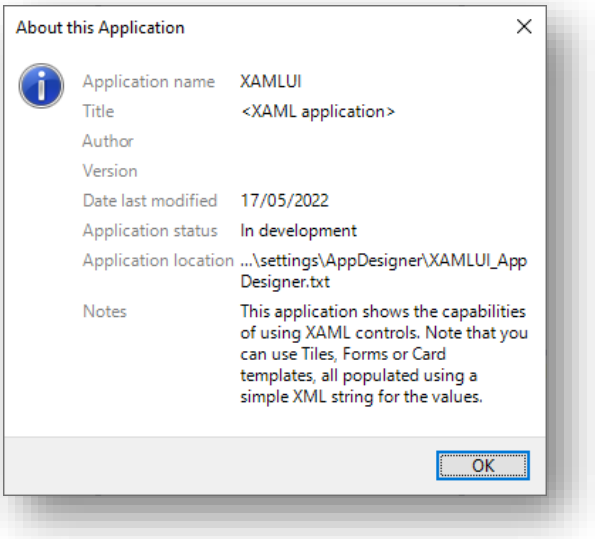
The following differences help a user identify a custom application:

1. The Windows icon (a blue rectangle) is different to a SYSPRO icon. This helps identify this application as a 'custom' one.

2. There is an **About this Application** option available from the Gear:



This shows information relevant to the current application:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Designing a Toolbar

The Toolbar control is the probably one of the most important controls of the Application Designer. It is the key UI component that the end user engages with.

Toolbar controls reside at the top of the main application pane, but toolbars can be created at the top of any pane if required and multiple toolbars can be defined in a single pane.

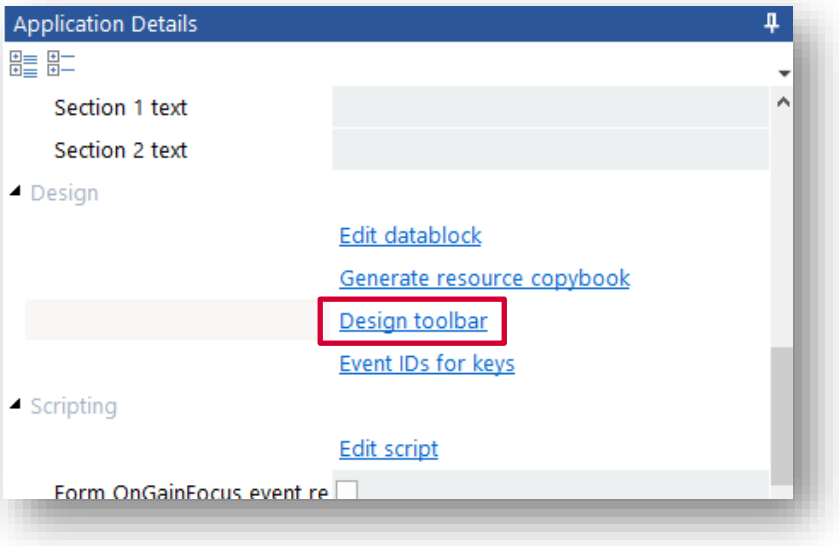
When a control is added to a toolbar, a unique Event ID is allocated to it.

When the control is used, an event is raised and the unique event id is passed back to the VBScript for processing along with relevant data, depending on the control type.

There are numerous Toolbar Callout functions that enable the VBScript to interact with the toolbar and its controls.

Creating and Editing Toolbars

To design a Toolbar, click the **Design toolbar** hyperlink:



To Edit a toolbar and change properties of buttons whilst in design mode, simply select the control from the toolbar and its properties will appear in the properties pane for editing.

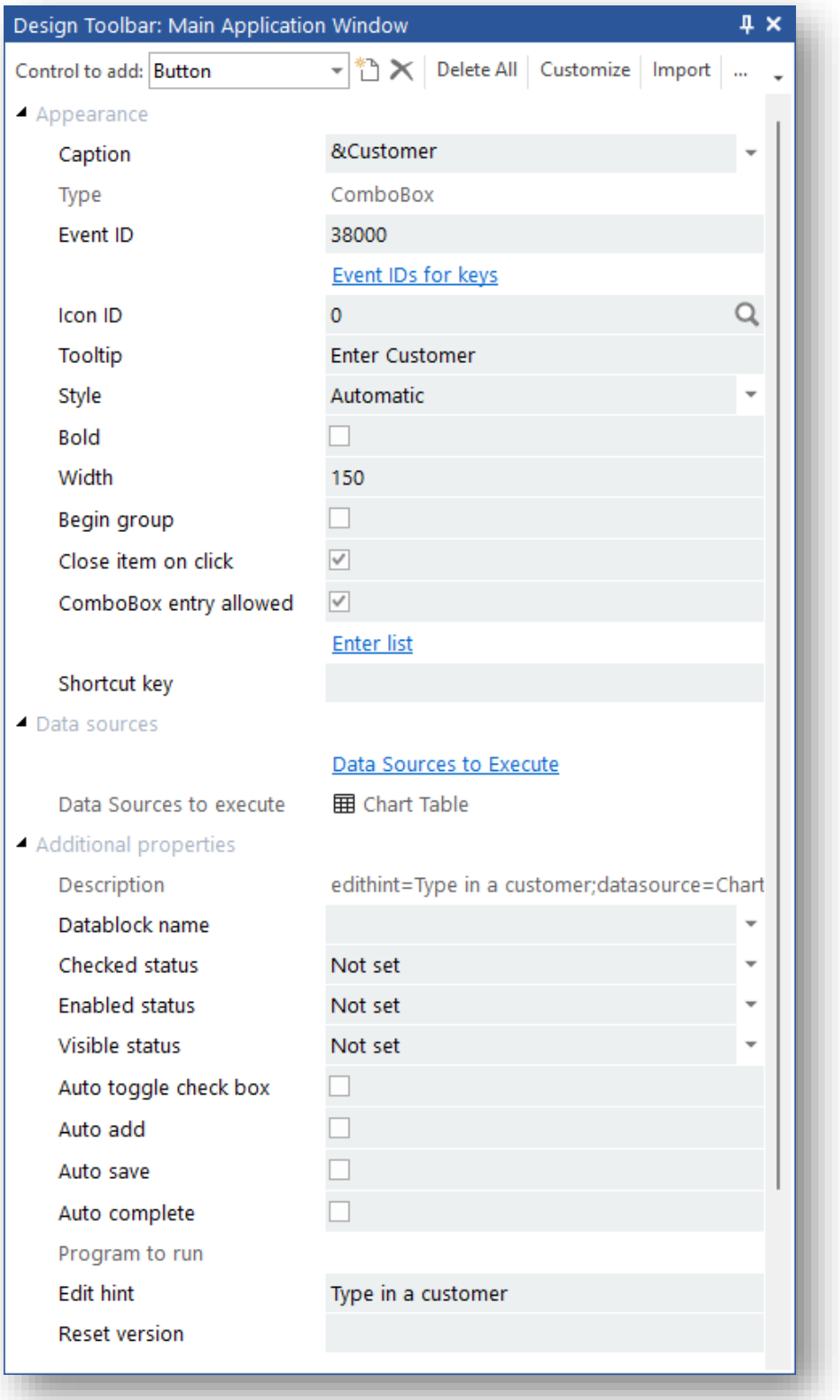
To re-organize the sequence of controls on the toolbar or move buttons to sub menus (i.e. Popup, Button popup, and Split popup buttons) click **Customize**. Move the Customize window away from the toolbar so it can be seen and then click and drag any control to a new position.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

GENERAL TOOLBAR FEATURES

Each control added to a toolbar is configured through the properties pane.

Here's an example of entering a **Customer** button:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

The HELP panel in the Application Designer explains what each option means. Associating a control with a Data Source is explained in the section Using Data Sources.

When typing in a caption, the Designer will automatically create a unique accelerator key. In the example below, an Ampersand (&) has been inserted before the first character.

When the Application runs, pressing the Alt key and the character together will have the same effect as selecting it with a mouse. The IDE will ensure that the ampersand (&) character is inserted before the first character that is not duplicated.

When creating a combo box (or Edit control) and a Caption is recognized as a SYSPRO key field name; then the Application Designer will allocate a standard event ID to it, which in turn links the standard SYSPRO browse icon to the combo box (and with it, all the functionality and search capability of the standard product).

When a control raises an event, the following data is passed to the VBScript:

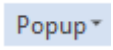
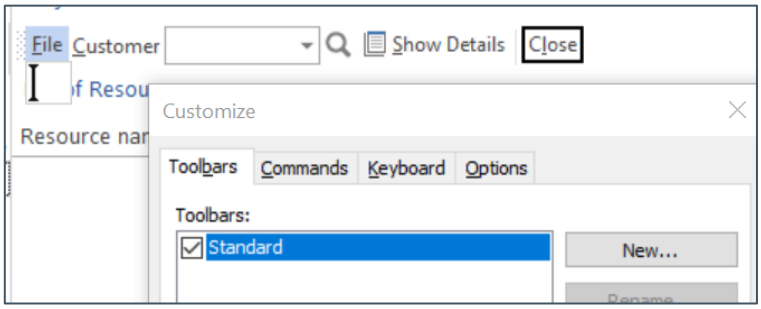
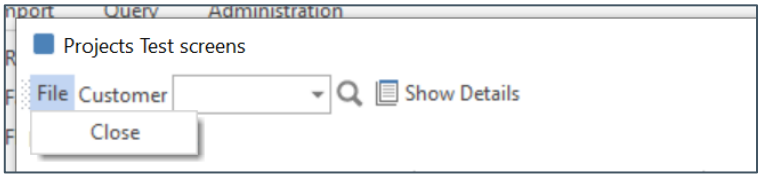
- **`EventType`** is of type 'ToolbarEvent'
- **`EventName`** is the name of the toolbar
- **`EventID`** is the unique ID allocated to the control
- **`Param1,2,3`** holds data depending on the control type

© SYSPRO. All Rights Reserved. All trademarks are recognized.

TYPES OF CONTROL

The following is a list of controls that can be added to a toolbar:

Note: When any control on the toolbar is executed, it raises a `ToolbarEvent` event type passing the **Event ID**.

Property	Explanation
Button	Traditional UI button. No values are included in `Param1,2,3`.
Button popup and Popup	Used as a "pull down menu". This control displays other toolbar controls such as buttons. It does not create its own event if pressed, it is just a container for other buttons. A Popup control differs slightly from a Button Popup in that it shows an expand button:  To set this up: 1. Create the popup. 2. On the toolbar, create each of the buttons that will reside in this popup. 3. Once the components have been created, use the toolbar customization screen (see above) to drag the child buttons into the popup button: 4. Go to customize toolbar and select child button, then move to the popup menu.  And when the App runs: 
Split button popup	This is a combination of a button and popup menu. No values are included in `Param1,2,3` when the button is pressed. If the down arrow is pressed, a sub menu of other controls is displayed. The popup menu behaves exactly as the Button popup.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Property	Explanation
Combo box	This is used to hold a list of items. The list of items can be static (i.e. entered at design stage and never changed) or dynamic (i.e. populated with data either from the VBScript or directly from SYSPRO). When an option is selected: ▪ **Param1** holds the item selected ▪ **Param2** holds the caption of the control
Edit	This is a text box that enables data to be entered into the toolbar. Entering data and pressing Return or Enter raises the event: ▪ **Param1** holds the data ▪ **Param2** holds the caption of the control
Checkbox	This is a single checkbox that can be ticked or not. ▪ **Param1** holds the check status. 0 indicates unchecked 1 indicates checked
Radio button	This is a single radio button that can be checked or not. ▪ **Param1** holds the check status. 0 indicates unchecked 1 indicates checked

© SYSPRO. All Rights Reserved. All trademarks are recognized.

CALLOUT FUNCTIONS

The list below indicates the callout functions that allow the VBScript to engage with the toolbar:

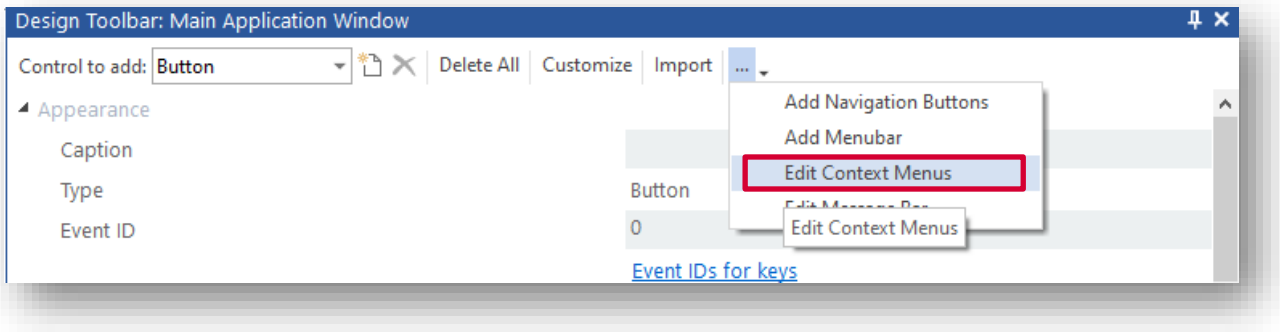
Toolbar callouts		
Set focus to a toolbar entry field	(ToolbarSetFocus, "xxxxxxxx", "nnnnn")	The toolbar field must be editable.
Set checkmark	(ToolbarSetCheckMark, "xxxxxxxx", "nnnnn")	
Unset checkmark	(ToolbarUnsetCheckMark, "xxxxxxxx", "nnnnn")	
Get checkmark state	(ToolbarGetCheckMark, "xxxxxxxx", "nnnnn")	Returns 1 (checked) or 0 (unchecked).
Toggle checkmark state	(ToolbarToggleCheckMark, "xxxxxxxx", "nnnnn")	Toggles the checkmark state (if checked, then set unchecked; if unchecked, then set checked).
Get value of a toolbar field	(ToolbarGetValue, "xxxxxxxx", "nnnnn")	Returns the value of the toolbar edit control. If it is a combo box control with enumerators, then the enumerator is returned.
Set value of a toolbar field	(ToolbarSetValue, "xxxxxxxx", "nnnnn|new value")	The toolbar field must be an edit control.
Set combo box index	(ToolbarSetIndexPointer, "xxxxxxxx", "nnnnn|index pointer")	This sets a specific list item by index number. The toolbar field must be a combo box control.
Set combo box index by value	(ToolbarSetIndexByValue, "xxxxxxxx", "nnnnn|enumerator")	This sets a specific list item using an enumerator. The toolbar field must be a combo box control.
Set tooltip of a toolbar field	(ToolbarSetTooltip, "xxxxxxxx", "nnnnn|tooltip")	
Change the status bar text	(ToolbarSetStatusBarText, "xxxxxxxx", "1|text")	Requires the section number (1 or 2) and the new text.
Disable a button	(ToolbarDisableButton, "xxxxxxxx", "nnnnn")	
Enable a button	(ToolbarEnableButton, "xxxxxxxx", "nnnnn")	
Add to end of dropdown list	(ToolbarAddToDropDown, "xxxxxxxx", "nnnnn|item caption")	The toolbar field must be a combo box control.
Set a dropdown list	(ToolbarSetDropDownList, "xxxxxxxx", "nnnnn|list")	The toolbar field must be a combo box control. The items in the list must be separated by a semi-colon.
Show a context menu	(ToolbarShowContextMenu, "xxxxxxxx", "nnnnn")	Returns the ID of the selected context menu item, or '00000' if no menu item selected.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Designing a Context Menu

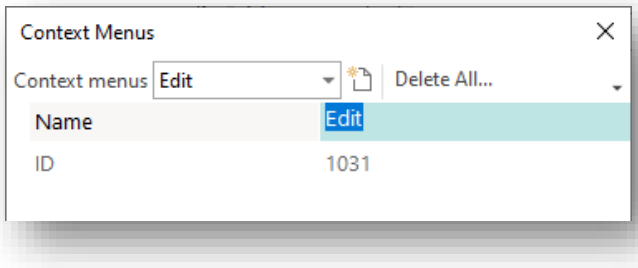
A context menu is a pop-up menu that provides shortcuts for actions.

A context menu is defined against the main application toolbar's **Menubar**.
To edit a context menu, you must first add a **Menubar** to the application toolbar, and then select **Edit Context Menus**:

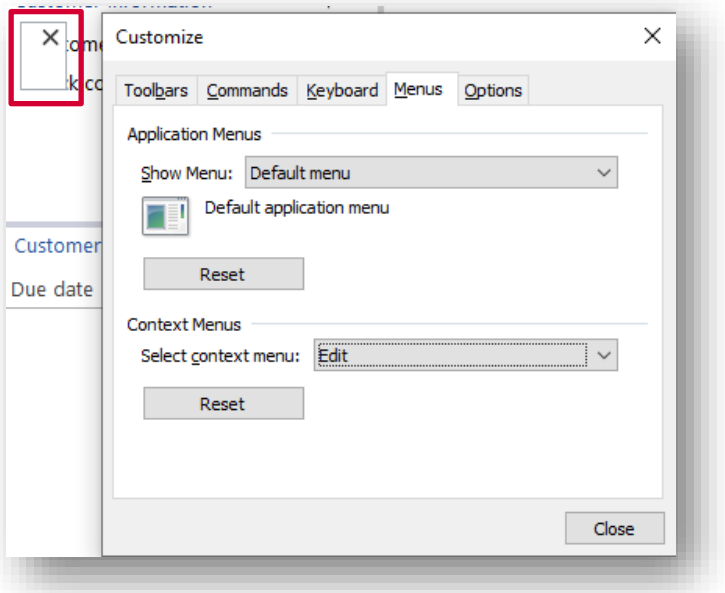


Add a context menu and define the **Name** for the menu.

Note the **ID** (e.g. 1031) of the context menu, as you will reference that when showing the context menu.

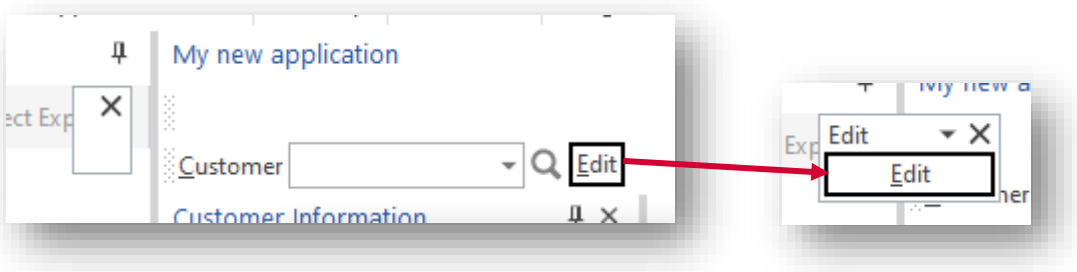


To add items to this context menu, click on **Customize** and then select the **Menus** tab. Then select the required context menu. The context menu is then displayed:



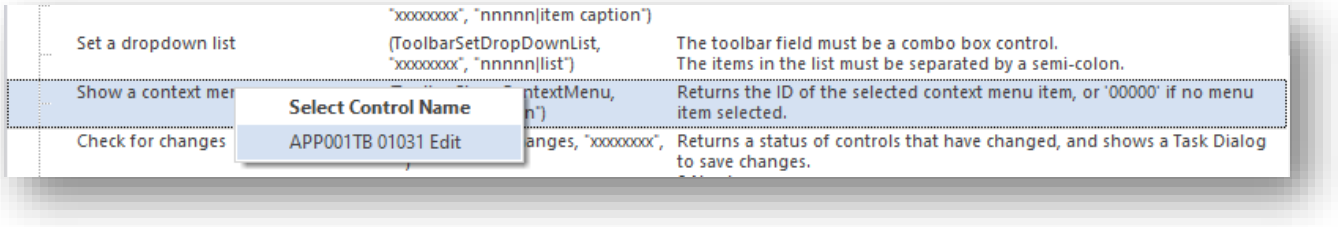
© SYSPRO. All Rights Reserved. All trademarks are recognized.

You can now drag any items from the main toolbar onto the context menu:



Close the Customize dialog box.

To show the context menu in your script, double-click the **Show a context menu** callout and select the context menu you wish to show:



If an item is selected from the context menu, its control ID will be returned in **`ReturnValue`**.

Considerations:

- You may have noticed that you must first define whatever buttons you want in your context menu on your main application toolbar.
- This technique will work both in SYSPRO and in Avanti.

CREATING TASK DIALOGS

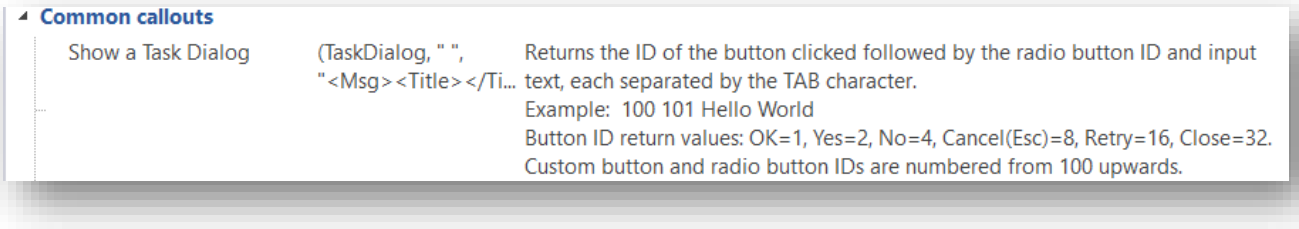
A task dialog is a dialog box that can be used to display information and receive simple input from the user.

The **`TaskDialog`** Callout is the Application Designer's equivalent to VBScript's MSGBOX command but is far more flexible, enabling the developer to provide **toast** notifications, traditional message box messages with buttons, text, and radio button interaction with the user.

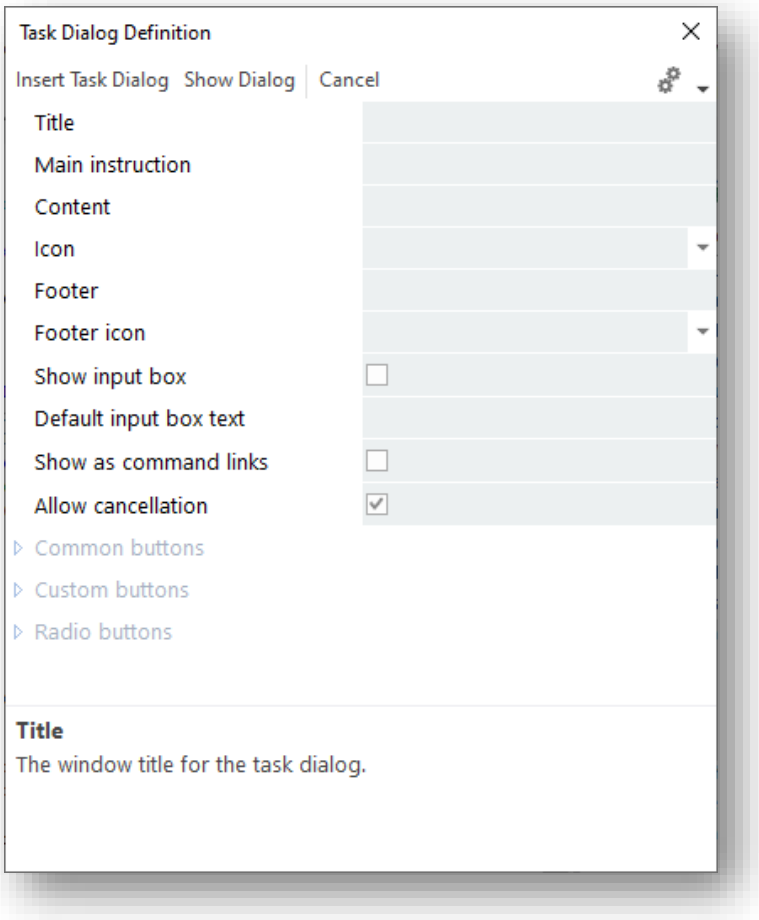
Its real flexibility comes when the different categories are mixed. For example, using standard and custom buttons with an input dialog. Or using standard and custom buttons with radio button dialog.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

The `TaskDialog` callout is found in the **Common callouts** section:



Double clicking **Show a task Dialog** will launch the Task Dialog Definition window, which helps create the VBScript to display a task dialog.



Click **Insert Task Dialog** to insert the Task Dialog code at the current script position.

Show Dialog shows a preview of the dialog.

Pressing any **Common buttons**, **Custom buttons** or **Radio buttons** will display the return values that will be sent to the VBScript. This enables testing of the dialog without having to run code.

The Task Dialog control will always return 3 variables delimited by a tab character:

- The Button ID
- The radio button ID
- The input text

© SYSPRO. All Rights Reserved. All trademarks are recognized.

If the variable isn't appropriate for the dialog, then "" is returned.

The return value can then be split into an array and each individual value can be accessed.

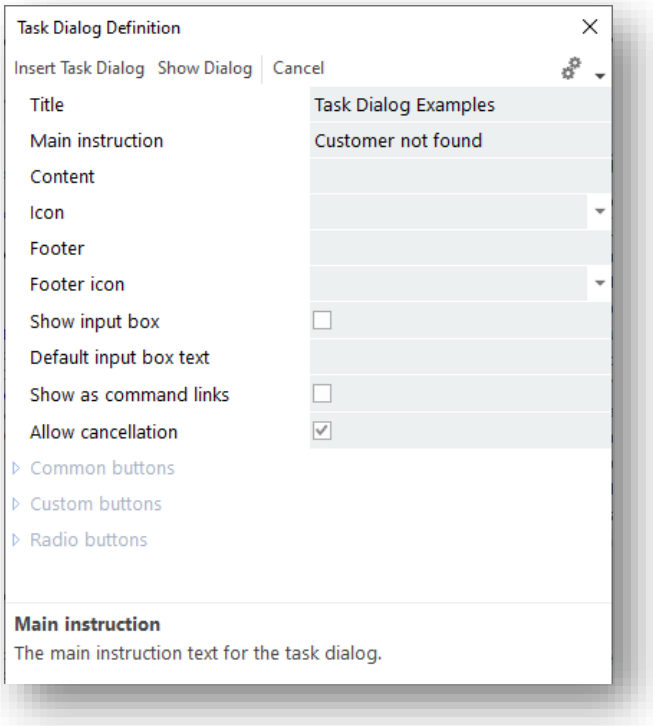
For example:

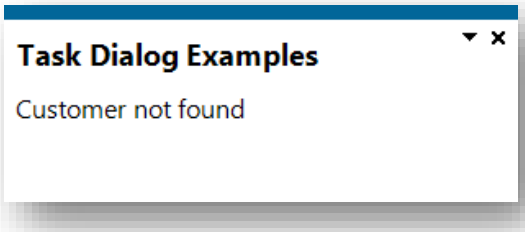
```
Function DisplayReturnValues(strRetVal)
dim i
dim returnValArray

returnValArray = split(strRetVal,vbTab)

'for i = 0 to ubound(returnValArray)
'    msgbox "i = " & i & " " & returnValArray(i)
'next

msgbox "Button ID: '" & returnValArray(0) & "'" & vbcrlf & _
       "Radio Button ID: '" & returnValArray(1) & "'" & vbcrlf & _
       "Input Text: '" & returnValArray(2) & "'", , "Task Dialog return values"


end function
```

Toast Notifications

Toast Notifications are popup dialogs that appear on the screen for a few seconds before disappearing automatically. They do not require any operator interaction and are useful to guide the operator and inform them of progress.

Defining a Task Dialog without any buttons or input box will automatically show a Toast Notification:

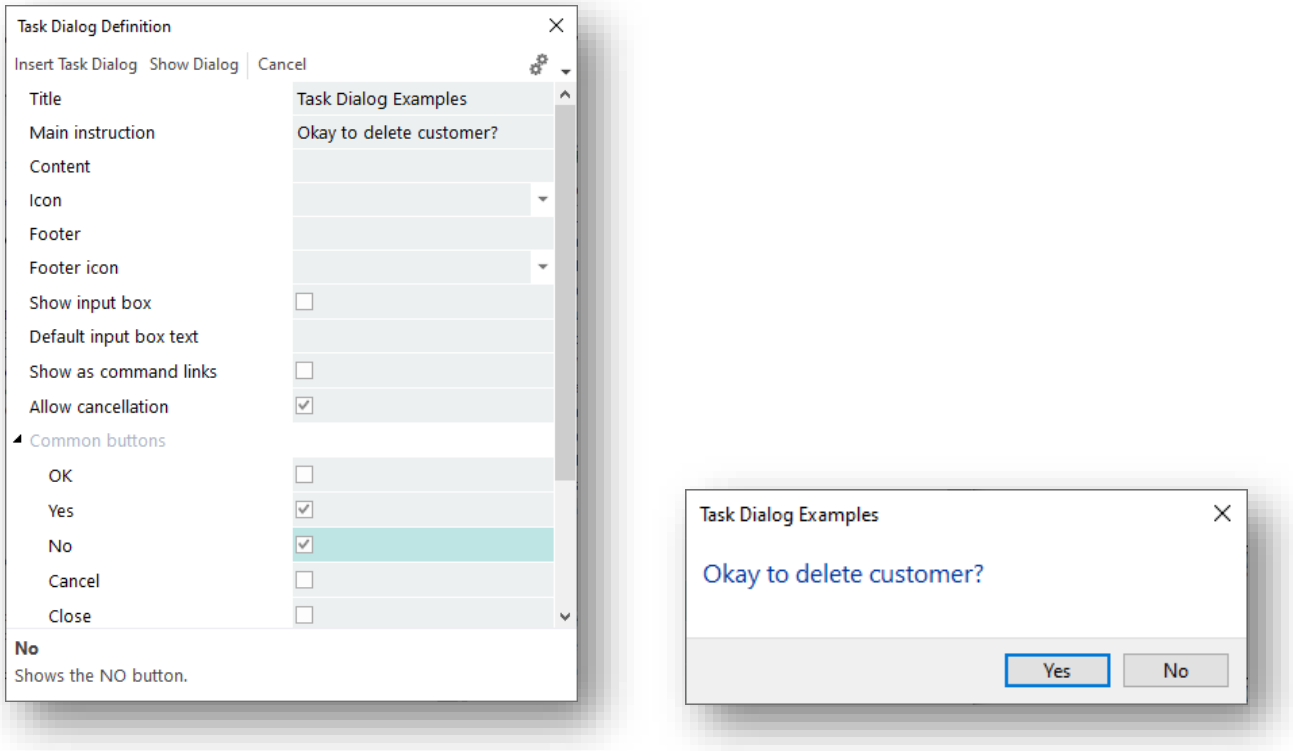




© SYSPRO. All Rights Reserved. All trademarks are recognized.

COMMON BUTTON TASK DIALOG

This form of the Task Dialog mimics the VBScript MSGBOX command:



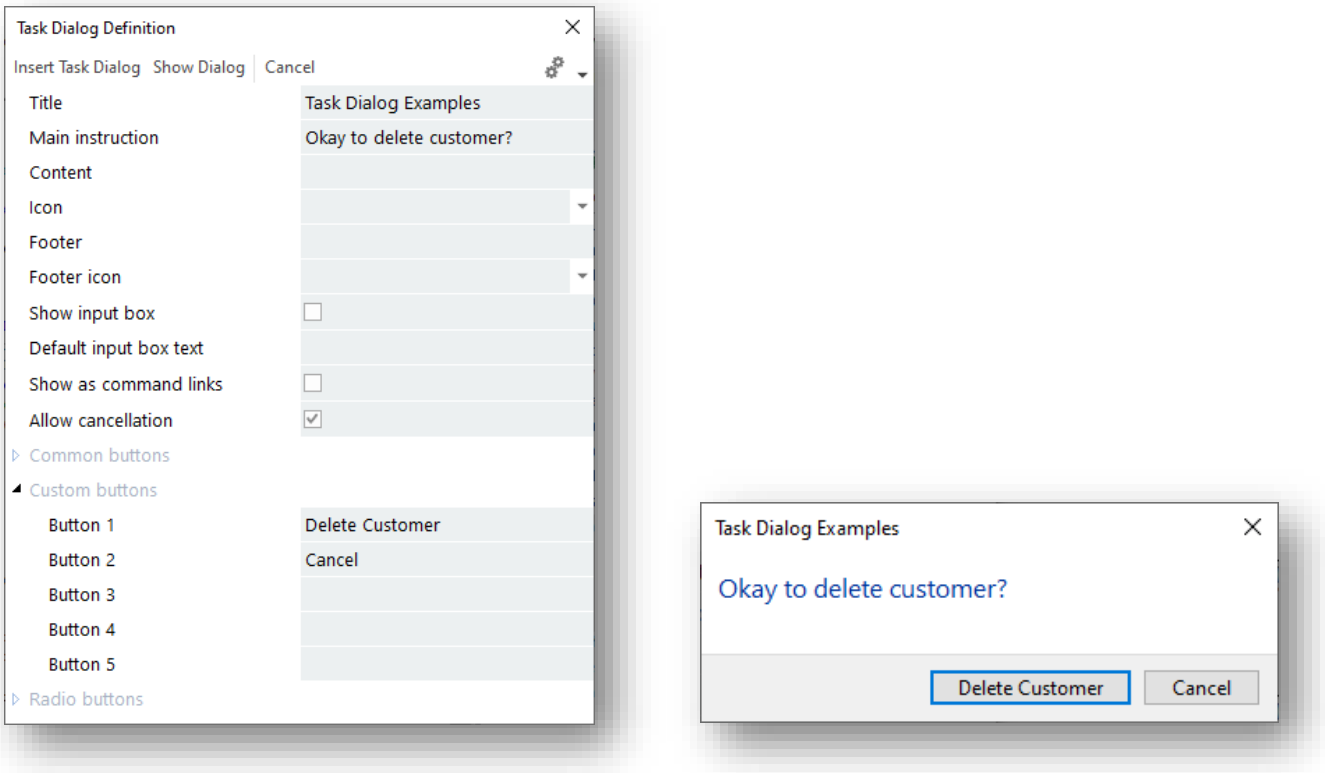
In this example, selecting **Yes** gives a return value of **002**, whereas selecting **No** returns **004**.

Note: The **Default button** property applies to the custom buttons as well.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

CUSTOM BUTTON TASK DIALOG

The custom button option allows the developer to set the caption of the buttons.



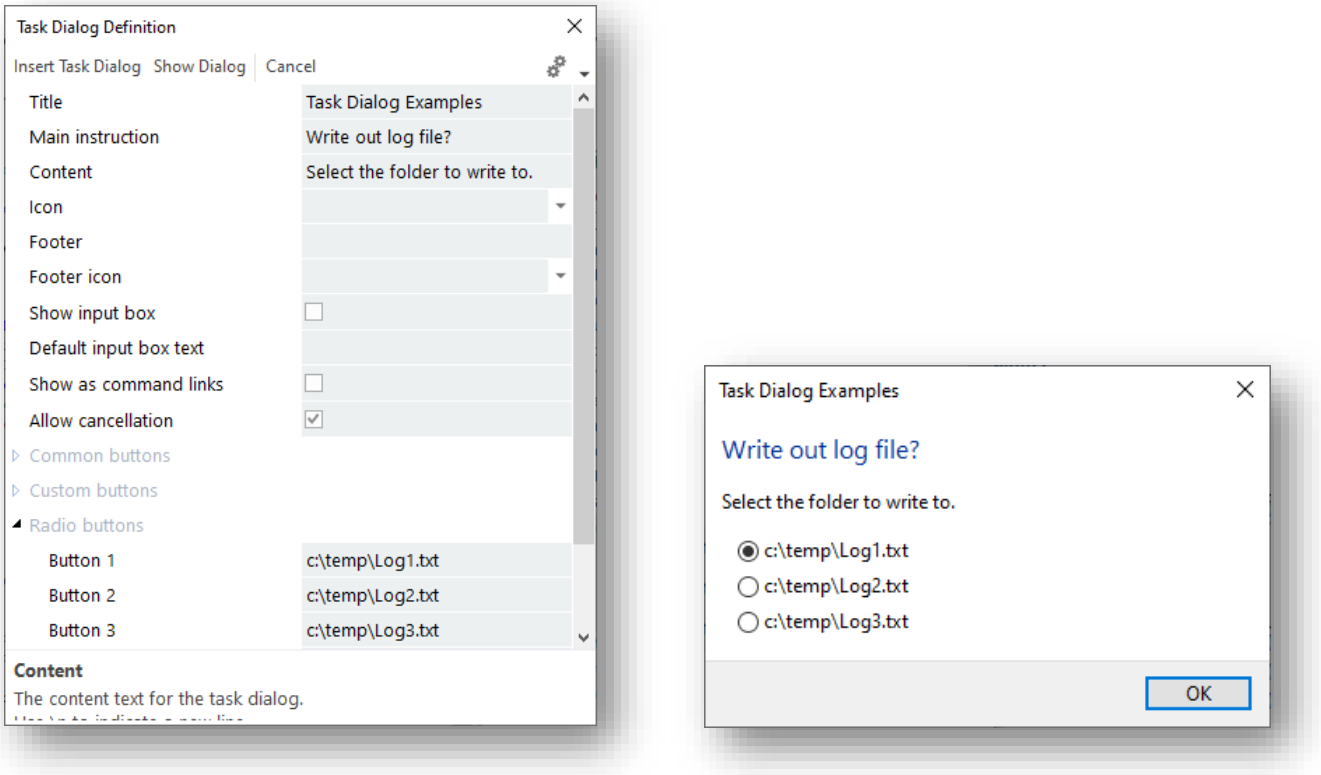
In this example, **Delete Customer** returns **100**. **Cancel** returns **101**.

If you need to set a default button, then use the **Default button** property in the common buttons section.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

RADIO BUTTONS

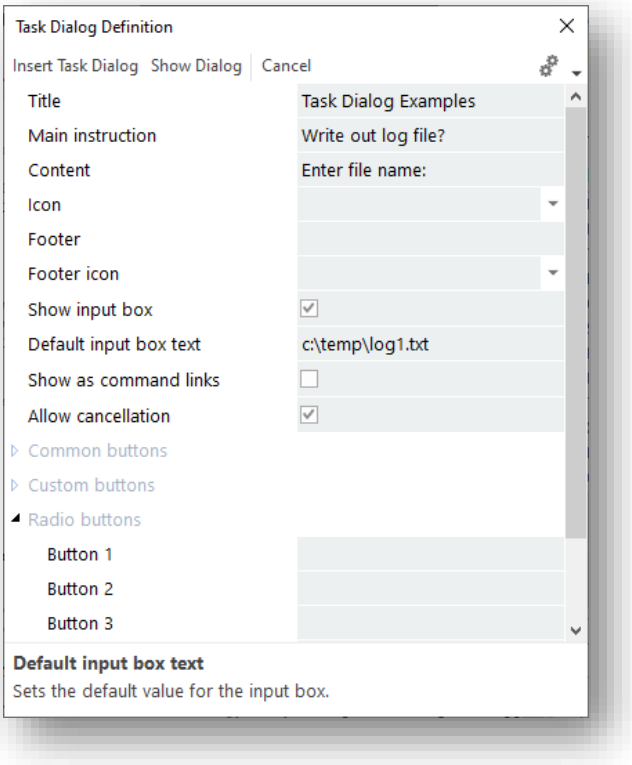
The radio button option provides a mechanism to mix buttons (both common and standard) with a radio button option:

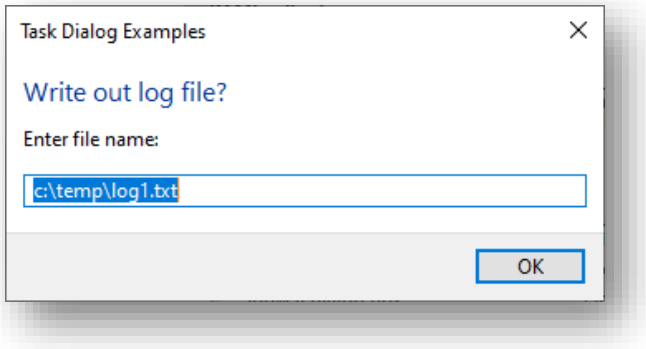


© SYSPRO. All Rights Reserved. All trademarks are recognized.

INPUT BOX DIALOG

The input box dialog enables entry of text into the dialog:

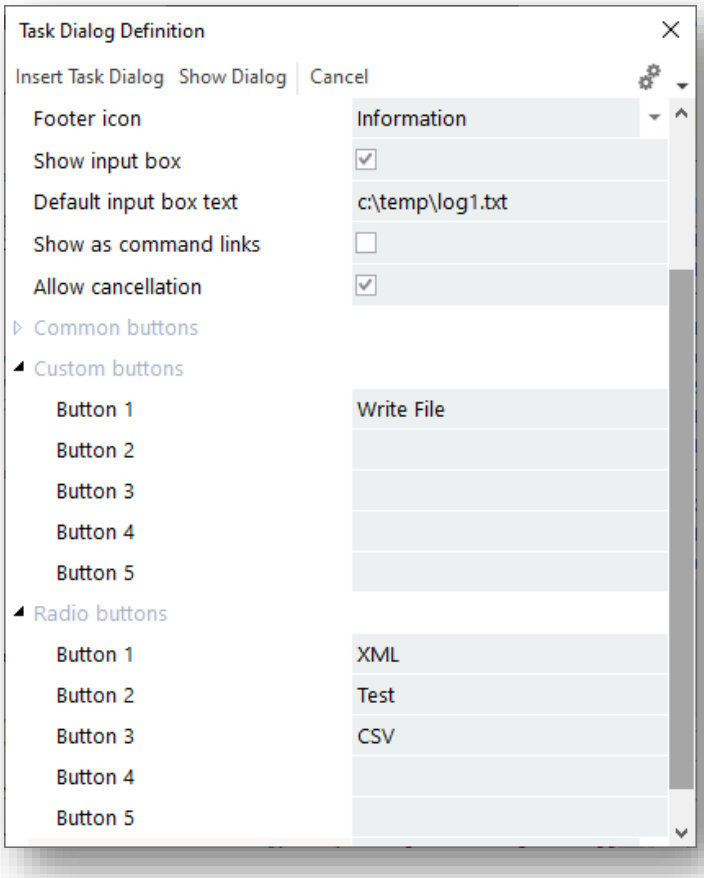


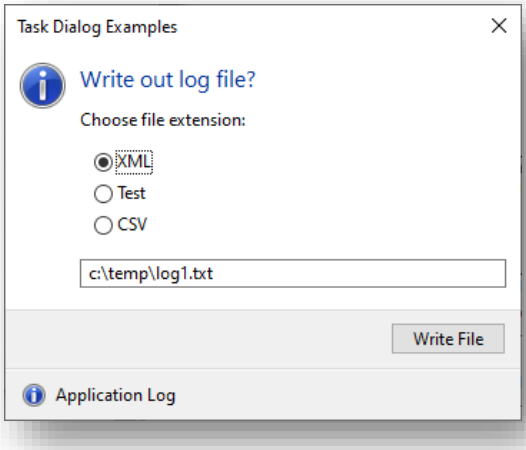


© SYSPRO. All Rights Reserved. All trademarks are recognized.

MIXING IT UP A BIT

It is possible to use properties from each of the sections and build a task dialog that is a combination of buttons, radio buttons and input text:





© SYSPRO. All Rights Reserved. All trademarks are recognized.

Creating Avanti Web Views

It is not possible to create an Avanti Web View directly without having first created the application in the Application Designer in SYSPRO. While the content in SYSPRO and in Avanti is generally the same, the layout of a Web View can be and is often different from a Windows Client User Interface.

For example:

Toolbar buttons are normally left-aligned when in SYSPRO, whereas in Avanti toolbar buttons can be left-aligned in the layout (for functional buttons) and right-aligned for options and Save buttons.

Perhaps worth noting is that a Web View can only render content that is already available in the SYSPRO application. For this reason, it is recommended that you first create the application and all the controls you may need before you design a Web View.

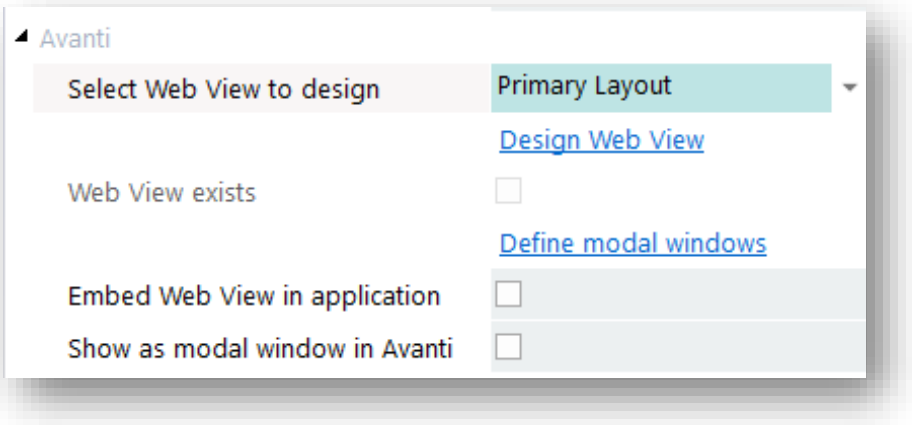
Having said this, you can always design and change the Web View whenever you want to add, change, or remove layout components.

Note: Certain controls are currently not supported in a Web View. These include:

- Flowgraph
- Image Viewer
- Document Viewer
- Tile
- Web Browser
- XAML Controls

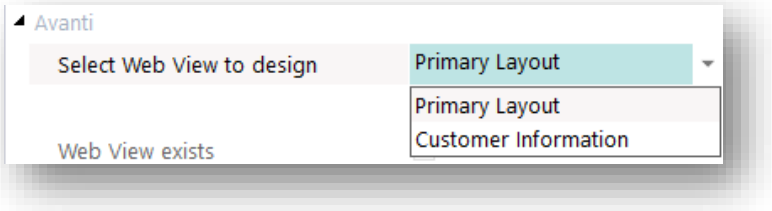
Designing a Web View

In the Application Details pane, you will see these options that are specific to Avanti:

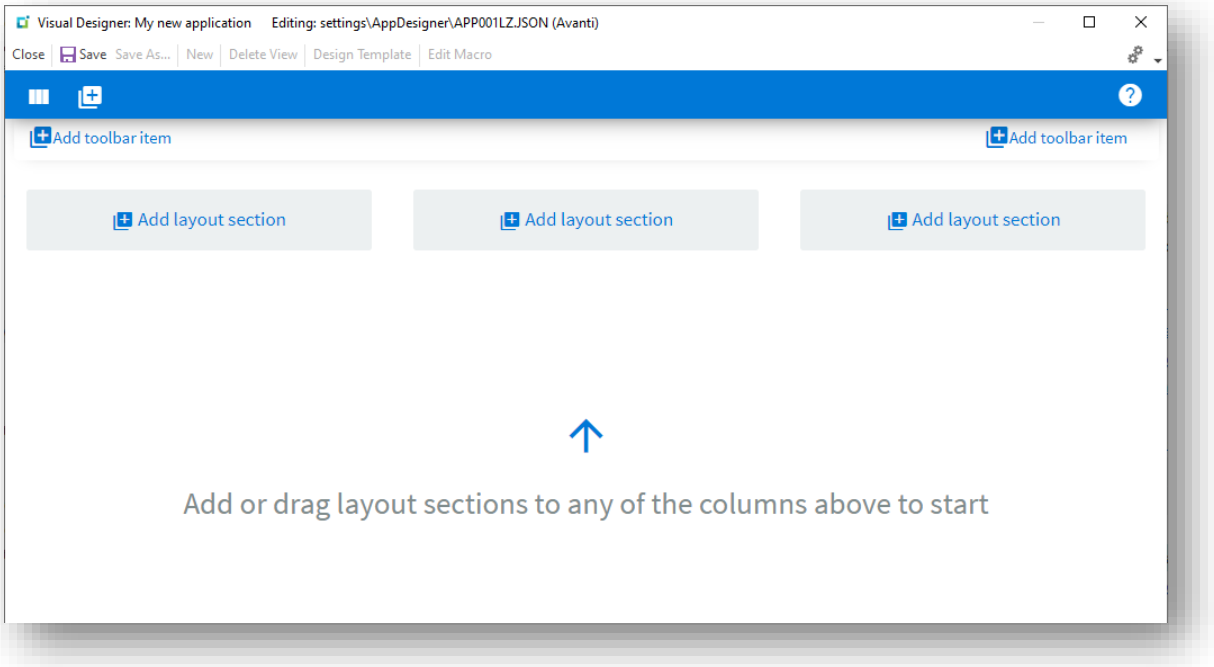


© SYSPRO. All Rights Reserved. All trademarks are recognized.

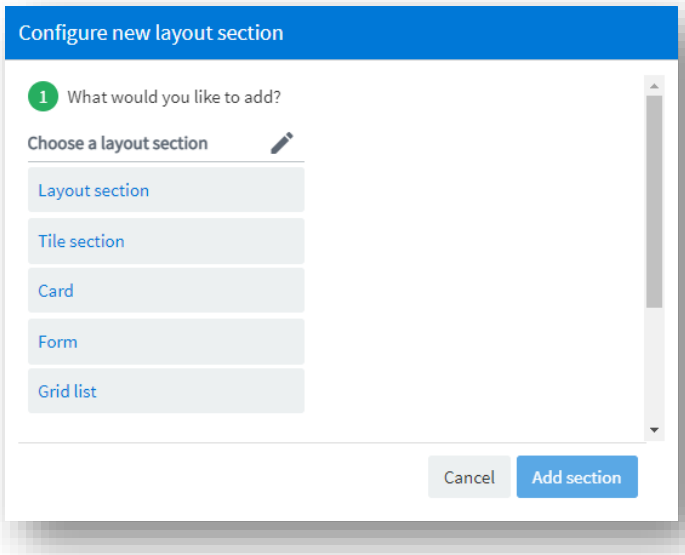
All Web Views have a 'primary layout' and optionally one or more modal windows. Any pane that is defined as a 'dialog box' will automatically be shown in the dropdown list, as will any additional modal window definitions.



Select the Web View you want to design and click on **Design Web View**. The Visual Designer is launched:

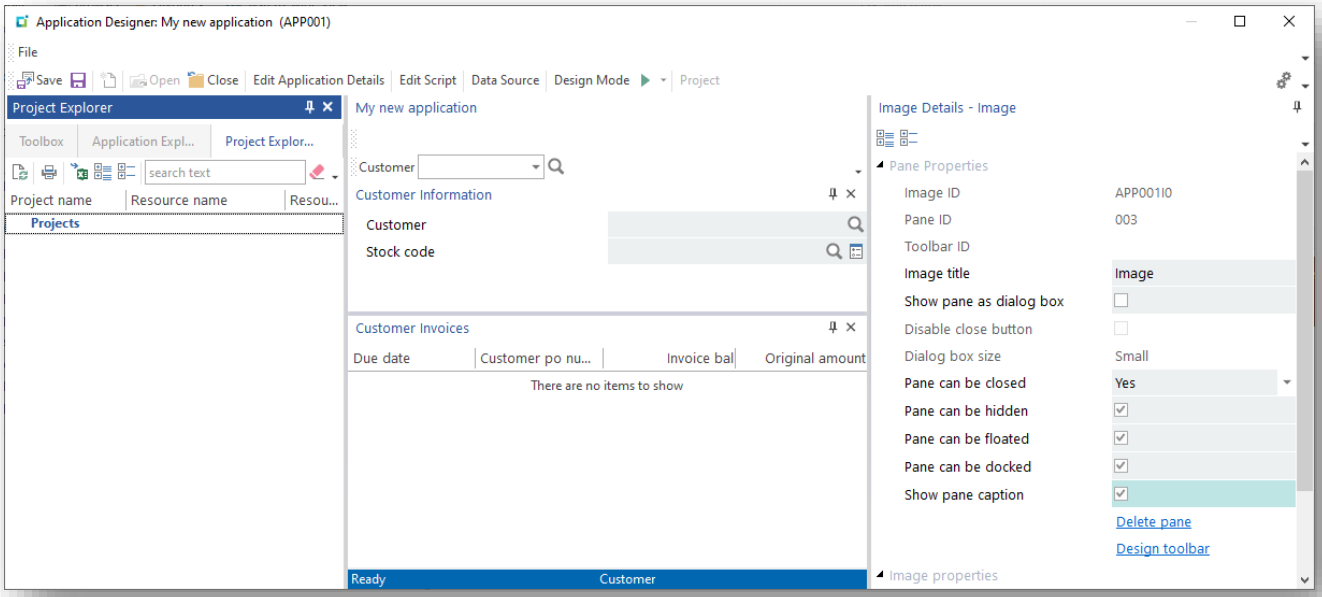


You can now design how you want your Web View to look by adding toolbar buttons and layout sections:

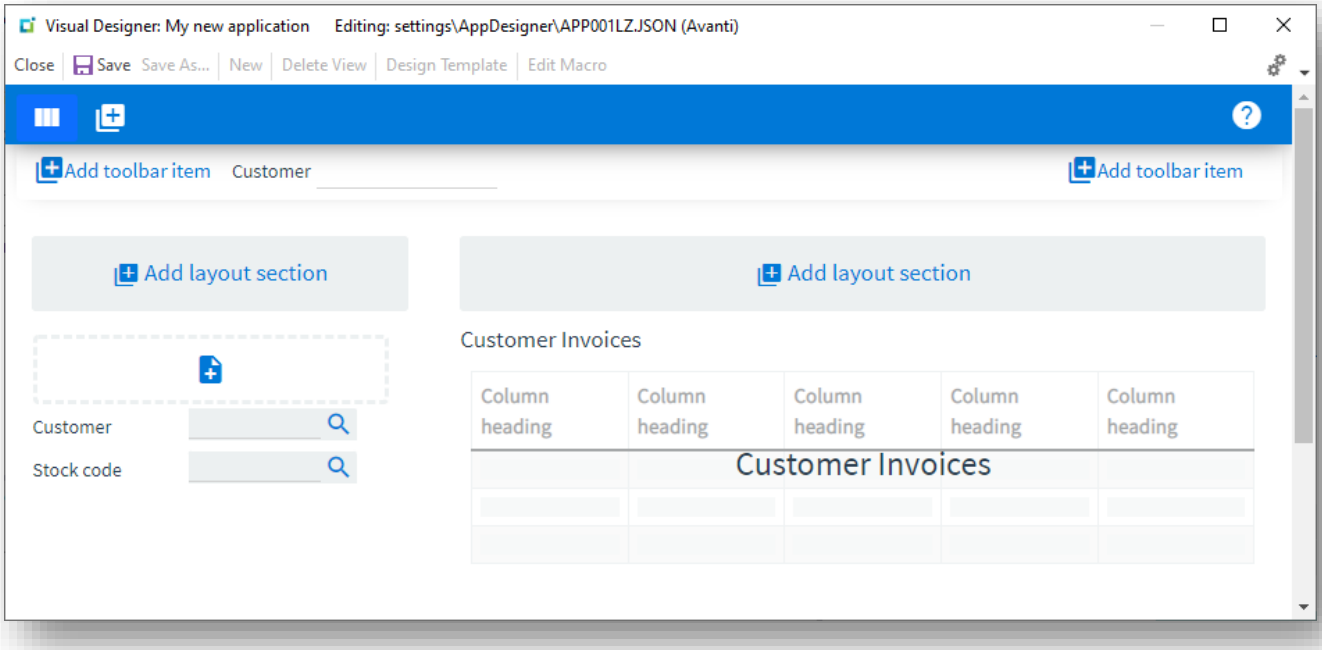


© SYSPRO. All Rights Reserved. All trademarks are recognized.

As an example, here is a simple application that has a toolbar, a form and grid:

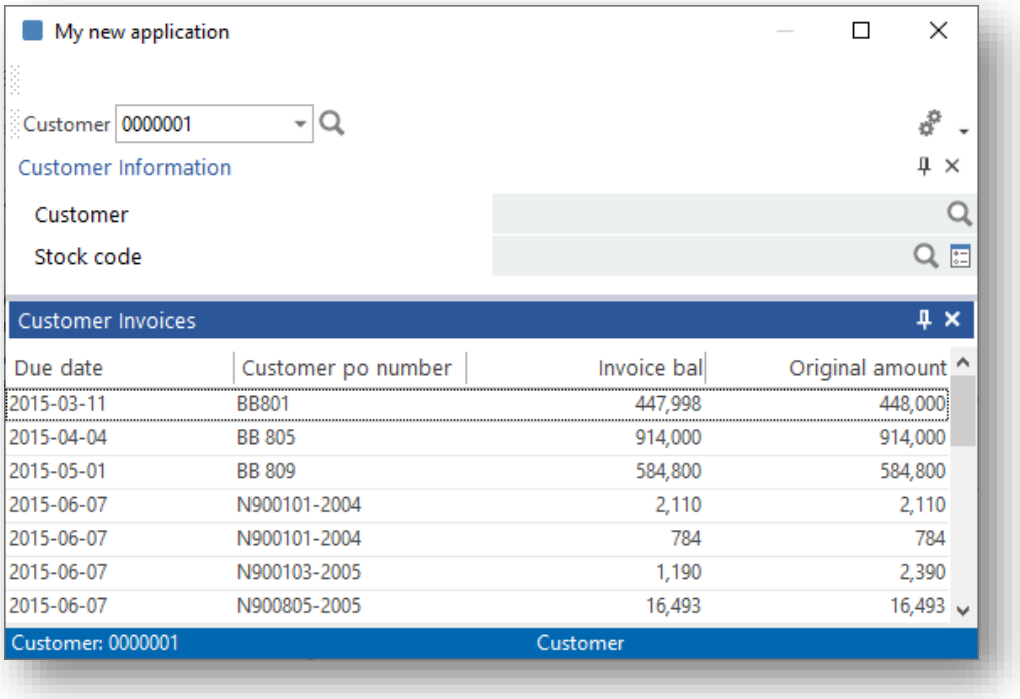


Here's how this application could be designed in the Visual Designer:

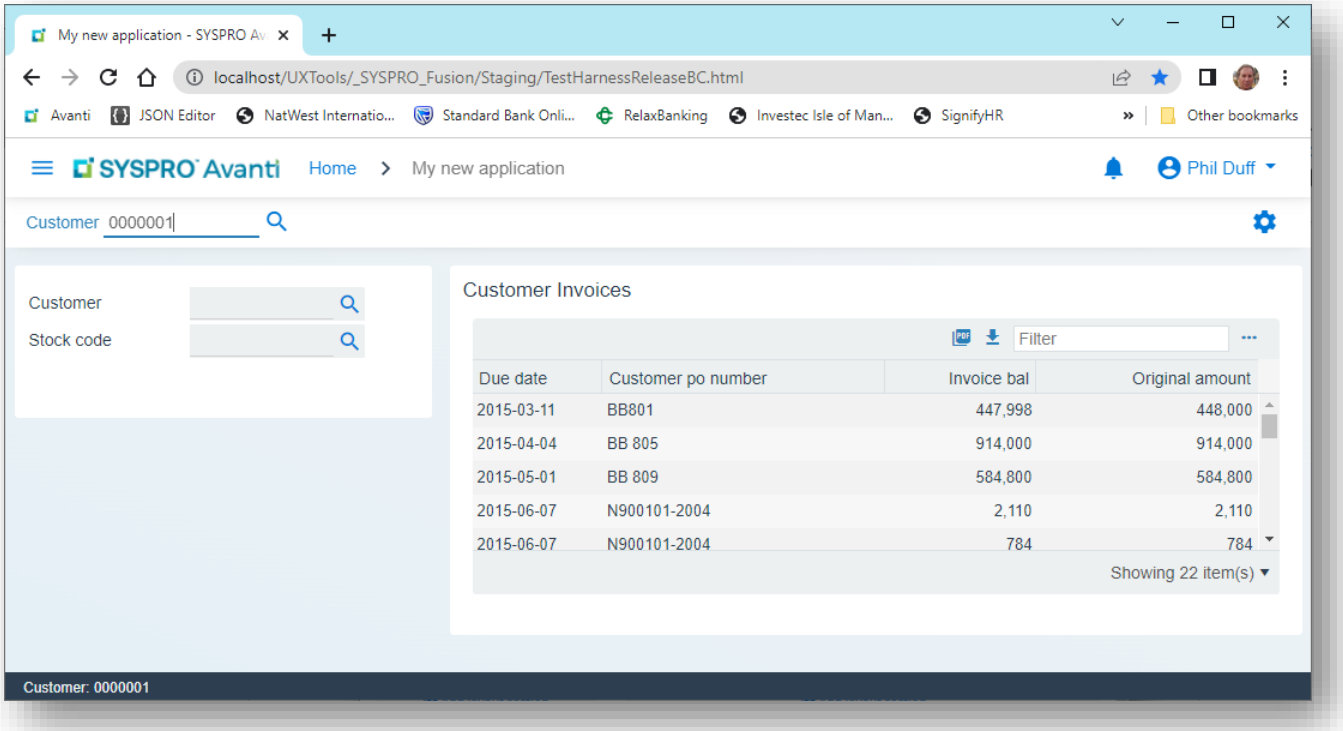


© SYSPRO. All Rights Reserved. All trademarks are recognized.

And here's what it looks like when running the application in SYSPRO:



And then in Avanti:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

SHOWING THE APPLICATION AS A MODAL WINDOW

By default, when an application loads, the window will be maximized in the web browser.

This has the advantage of being able to use the full width and height of the web browser. However, if the application's initial window size is defined as 'small', then it might be appropriate to enable the option **Show as modal window in Avanti**.

SHOWING MODAL WINDOWS

When a Dialog Box is shown using the **`ShowDialogBox`** function, the IDE will automatically issue an Avanti **`ShowModalWindow`** statement.

When the Dialog Box is closed the Avanti, the **`CloseModalWindow`** statement is executed.

Thus, Dialog boxes in SYSPRO work identically in Avanti.

Note: The name of the Web View for a dialog box will be generated as `UX_APP_`*`NameOfControl`*`.JSON` and saved in the `Settings\Appdesigner` folder.

FORM PROPERTIES

In the Form details pane, there is an option **Disable auto save**.

This option should be checked if the form's values should not be automatically saved when running in Avanti.

Examples of ensuring this is checked are month-end programs (e.g. ARSP01, etc.) as these types of programs should not have form values automatically saved upon exiting the application.

DESIGNING LIST VIEWS FOR AVANTI

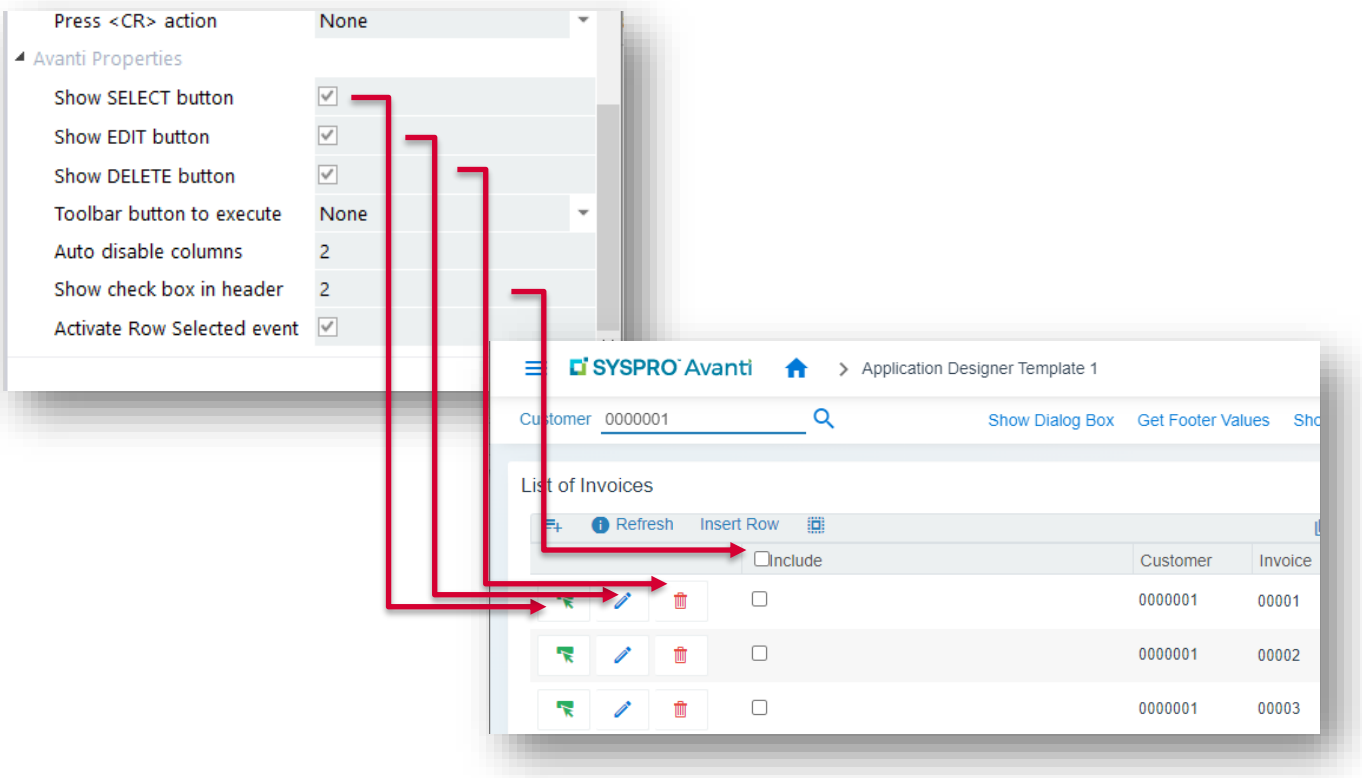
In the List view details pane, you will find the **Avanti Properties**.

These options further control the behavior specifically for an Avanti grid:

Property	Explanation
Show SELECT button	Shows a SELECT button.
Show EDIT button	Shows an EDIT button.
Show DELETE button	Shows a DELETE button.
Toolbar button to execute	The toolbar button to execute when the DELETE button is clicked.
Auto disable columns	Columns to be automatically disabled when cell value changes.
Show check box in header	Show check box in column header(s).
Dynamic combo box columns	Columns that contain dynamic combo boxes.
Hyperlink columns	Columns that contain hyperlinks.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

For example:



CREATING ADDITIONAL MODAL WINDOWS FOR AVANTI

A modal window is automatically shown in Avanti for any pane within an application that is defined as a dialog box when the callout **`ShowDialogBox`** is executed. However, there are times when you need to show a modal window in Avanti that's not specifically tied to a dialog box.

Typically, this is when the layout in a Web View is of necessity quite different to that in the SYSPRO Windows version.

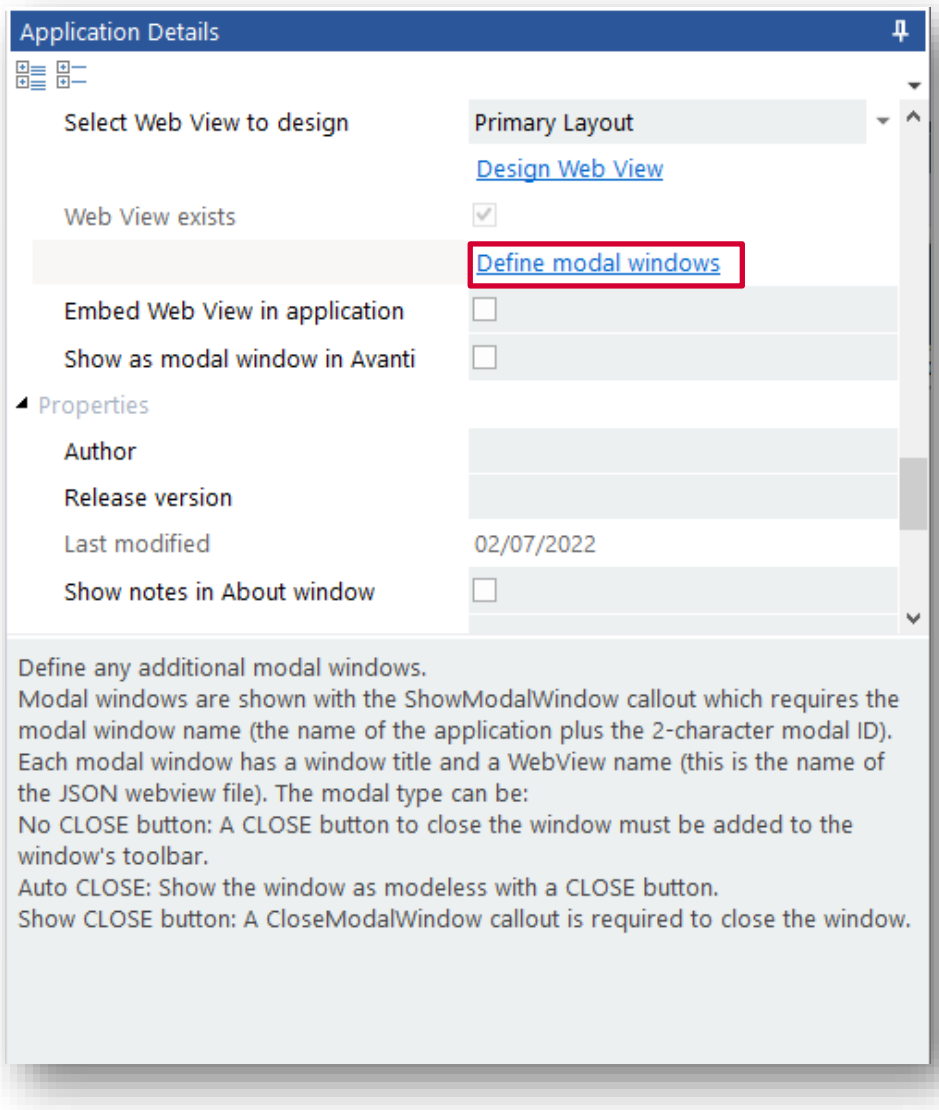
To create additional modal windows and have them shown at appropriate times when running an application requires the following:

1. Define the modal window properties in the Application Designer.
2. Design the Web View for the modal window.
3. Show the modal window using the callout **`ShowModalWindow`**.
4. Close the modal window using the callout **`CloseModalWindow`**.

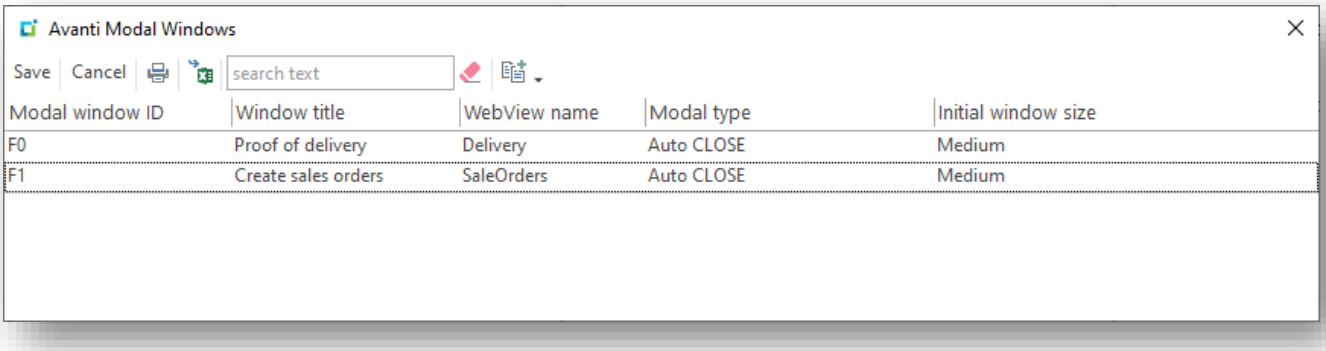
© SYSPRO. All Rights Reserved. All trademarks are recognized.

DEFINING MODAL WINDOWS

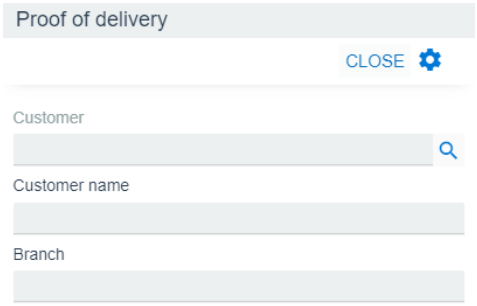
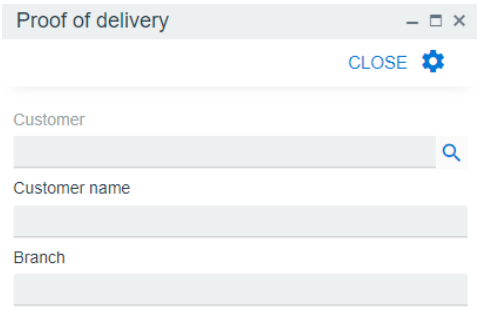
In the **Application Details** properties, click on the link **Define modal windows**:



Enter up to 5 modal windows:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

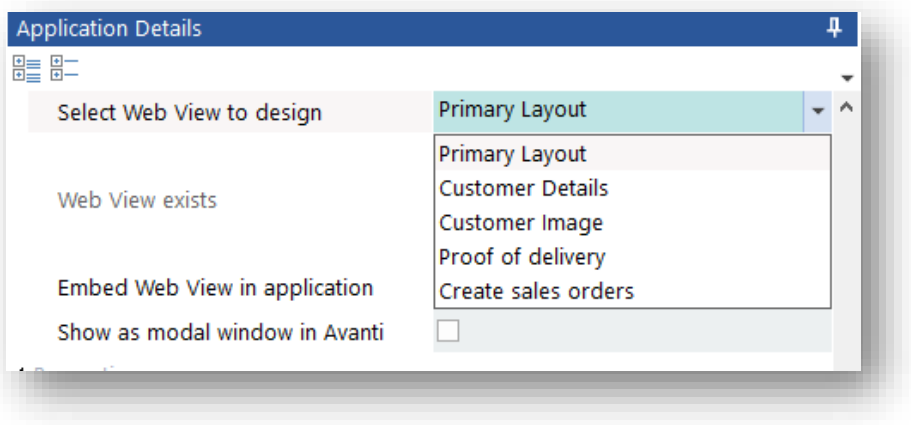
Column	Explanation
Modal window ID	The **Modal window ID** must be a 2-character ID. When you show a modal window using the `ShowModalWindow` callout, you must specify the 'modal name' which comprises the application name and modal window ID. Therefore, once you have used the `ShowModalWindow` in your code, you must change your code if you subsequently change this modal window ID.
Window title	The **Window title** must be non-blank and is shown as the title of the modal window.
Web View name	The **Web View name** defines the name of the Avanti JSON file associated with this modal window. It must be non-blank and cannot contain any special characters (/ ? : & \ * " < > | # % or space).
Modal type	The **Modal type** describes how the modal window is to be 'closed': Modal windows can be shown with or without a CLOSE button (i.e. the **X** at the top right-hand corner of the window). The modal type may be one of the following: ▪ **NoCloseButton** No CLOSE button is shown. The only way to close this type of modal window is to issue the `CloseModalWindow` callout. This is used when some logic needs to be performed before the window is closed. Note that the window is shown 'modal':  ▪ **AutoClose** A CLOSE button is shown. Clicking on the **X** will automatically close the window. The window is shown 'modeless', meaning that the user can click behind the window: 

© SYSPRO. All Rights Reserved. All trademarks are recognized.

	▪ **ShowCloseButton** A CLOSE button is shown. Clicking on the **X** will trigger the event **`ModalWindowClose`** and then the **`CloseModalWindow`** callout can be executed to close the window. This is used when some logic needs to be performed before the window is closed. Note that the window is shown 'modal'.
Initial window size	The **Initial window size** may be Small, Medium or Large and specifies the default window size when it is initially shown.

Designing a Web View for the Modal window

To design a Web View for a modal window, select the modal window from the drop-down list in **Select Web View to design** and then click **Design Web View**.



Showing or closing a modal window

The additional modal windows are not automatically shown. You must decide in your code the appropriate time to show a modal window.

Use the callout **`ShowModalWindow`** with the modal name comprising of the 6-character application name and the 2-character modal ID.

Unless the modal window is defined as **AutoClose**, you will have to issue the **`CloseModalWindow`** callout in your code to close the modal window using the same modal name.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Using Data Sources

The purpose of data sources is to allow you to create applications that obtain data from a variety of sources, and optionally bind data to Forms, List views or Charts. All this without having to do any code.

Data sources generally work as follows:

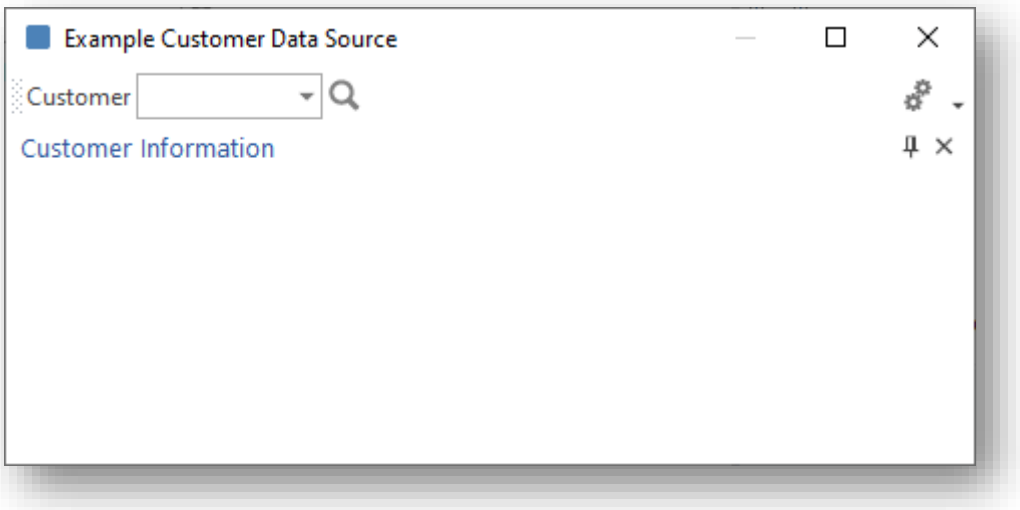
1. You create a data source that points to a SYSPRO table, SYSPRO business object or Custom data source.
2. You add fields to a form (or columns to a list view or data points for a chart) and in so doing define which data source is to be applied.
3. You decide at which point in the application that one or more data sources should be executed.
   It's important to note that most data sources (if not all) would require some sort of key (a Customer number, for example) to be applied.

Creating a Data Source

Let's see how this works in an example:

In an application, you enter a Customer number in the main toolbar and then we want various form fields to be automatically updated with information derived from the Customer master table.

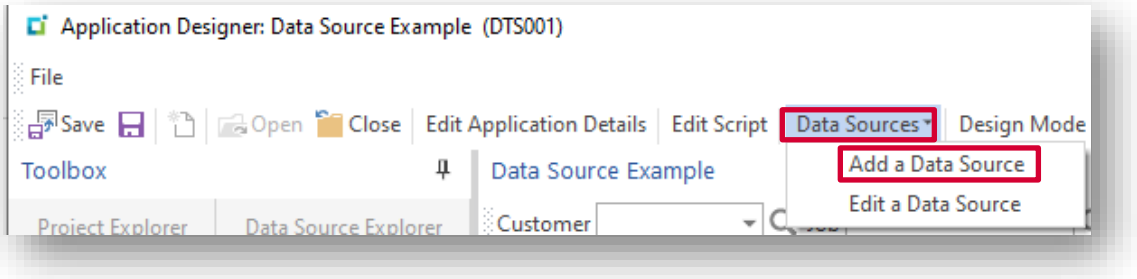
To simplify this explanation, let's assume we have this outline application that contains a Customer toolbar field and a blank form:



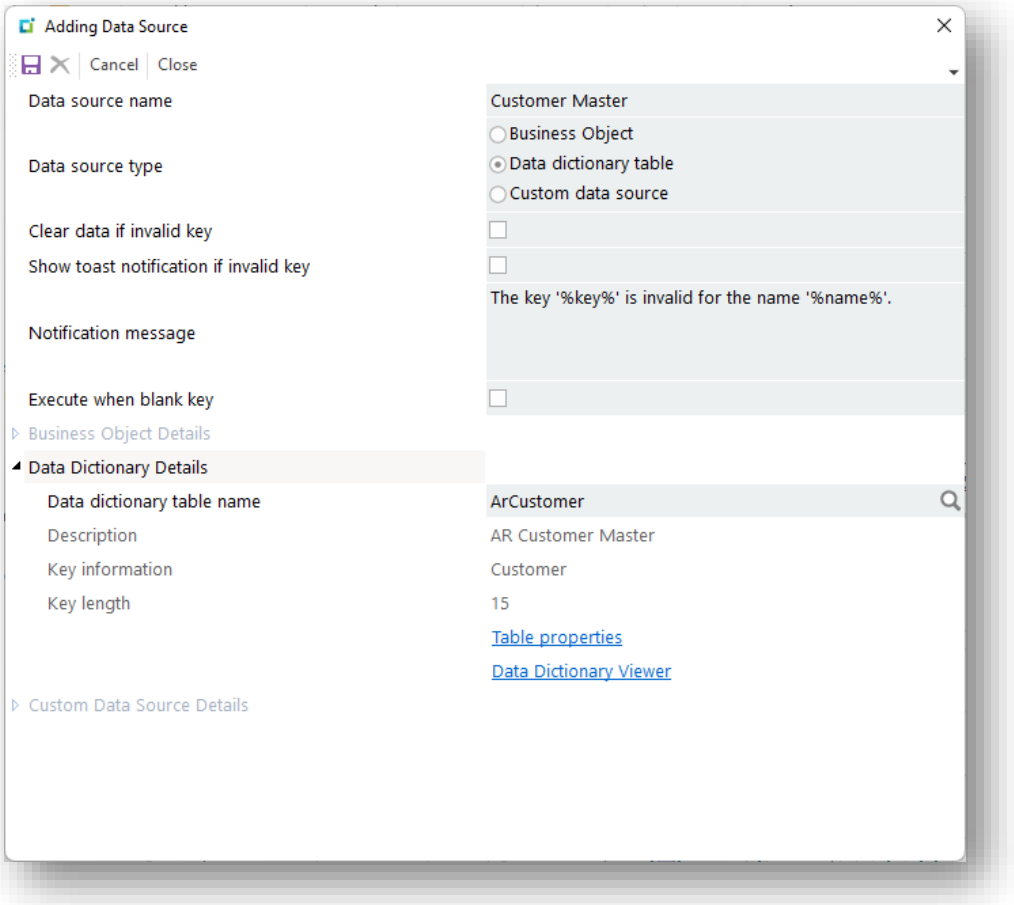
© SYSPRO. All Rights Reserved. All trademarks are recognized.

STEP 1: CREATE A DATA SOURCE

From the toolbar, click Data Sources > Add a Data Source:



The data source window is displayed:



Each data source must have a unique name by which it is identified.

- This can be up to 30 characters.
- Once created, you cannot change the name of a data source.
- In the example above, we have entered **Customer Master**.

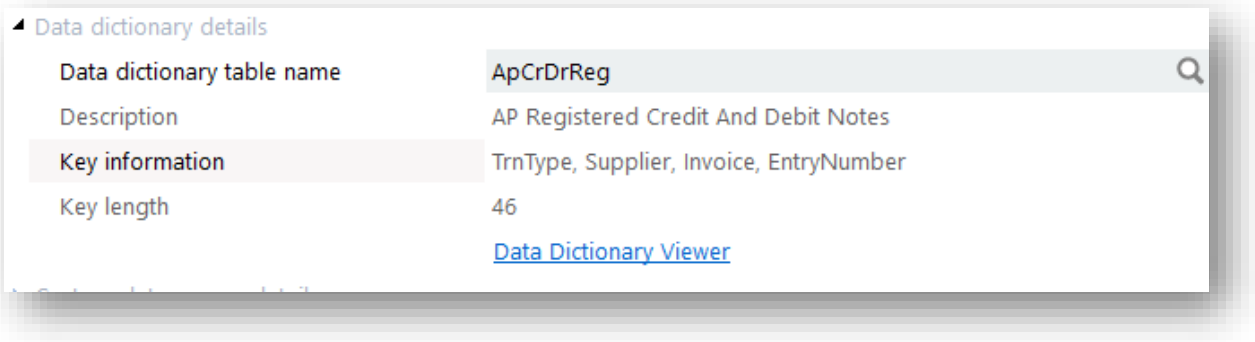
There are various types of data sources (Data dictionary table name, Business Object, Custom data source).

- For this exercise we have selected **Data dictionary table Name**.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

In the **Data dictionary table name** field, you can start typing the name and predictive search will help you identify the correct table. You can also press the browse button to see all available tables.

- In the example above, we have entered **ArCustomer**.
- Some tables may require a key constructed of multiple elements. This information is shown in the Key information property as column names separated by a comma:



  You can also click on **Data dictionary viewer** to view all tables and their properties.

Optional configuration for the scenario of an invalid key being entered:

- You can define if your form field values should automatically be cleared.
- You can indicate if a toast notification should be displayed and define the contents of the Notification message.
  - The message can optionally contain the placeholders **`%key%`** (the key supplied to the data source) and **`%name%`** (either the Table or Business Object used in the data source).
  - This message is shown only if the option **Show toast notification if invalid key** is switched on.

Execute when blank key. By default, this option is unchecked. This indicates that the data source will not be executed if the KEY value to be applied is blank. You will need to check this option if a data source can be executed without a KEY having to be supplied.

You can change these options later if required.

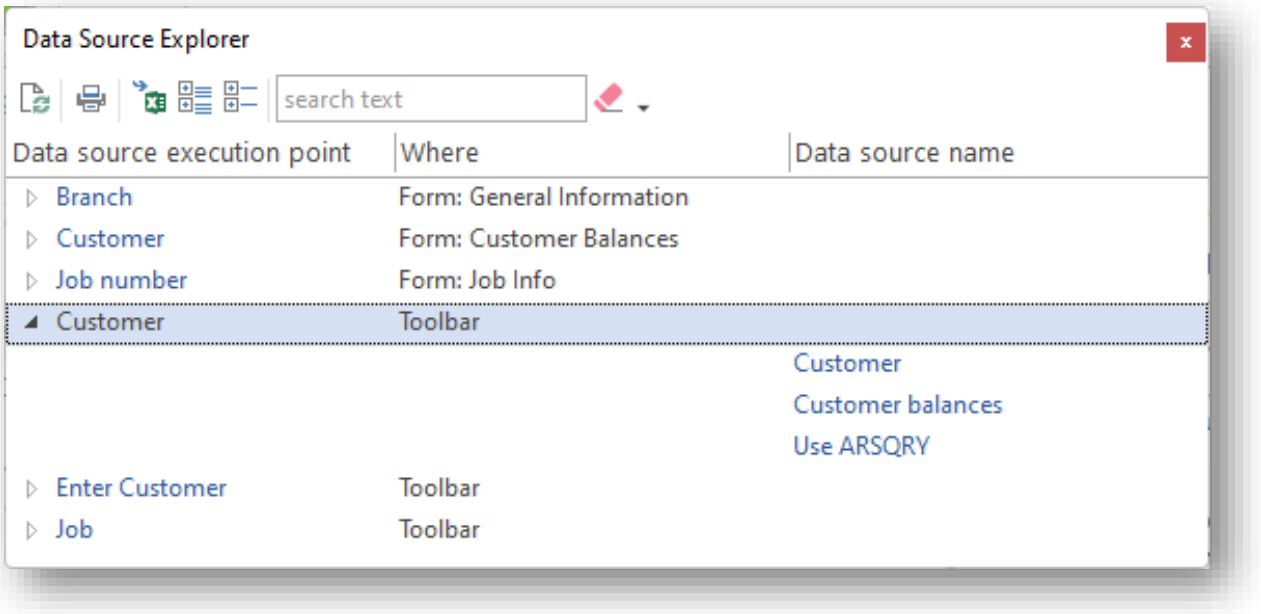
Note: When a data source is executed after a toolbar button, form field or list view cell value has changed (and if the KEY supplied to the data source is invalid) then focus will be automatically set to that toolbar, field or cell. This helps the user re-enter a valid key.

However, this doesn't apply to a Data Source that is executed from a script callout.

Click the **SAVE** button, followed by the **CLOSE** button.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

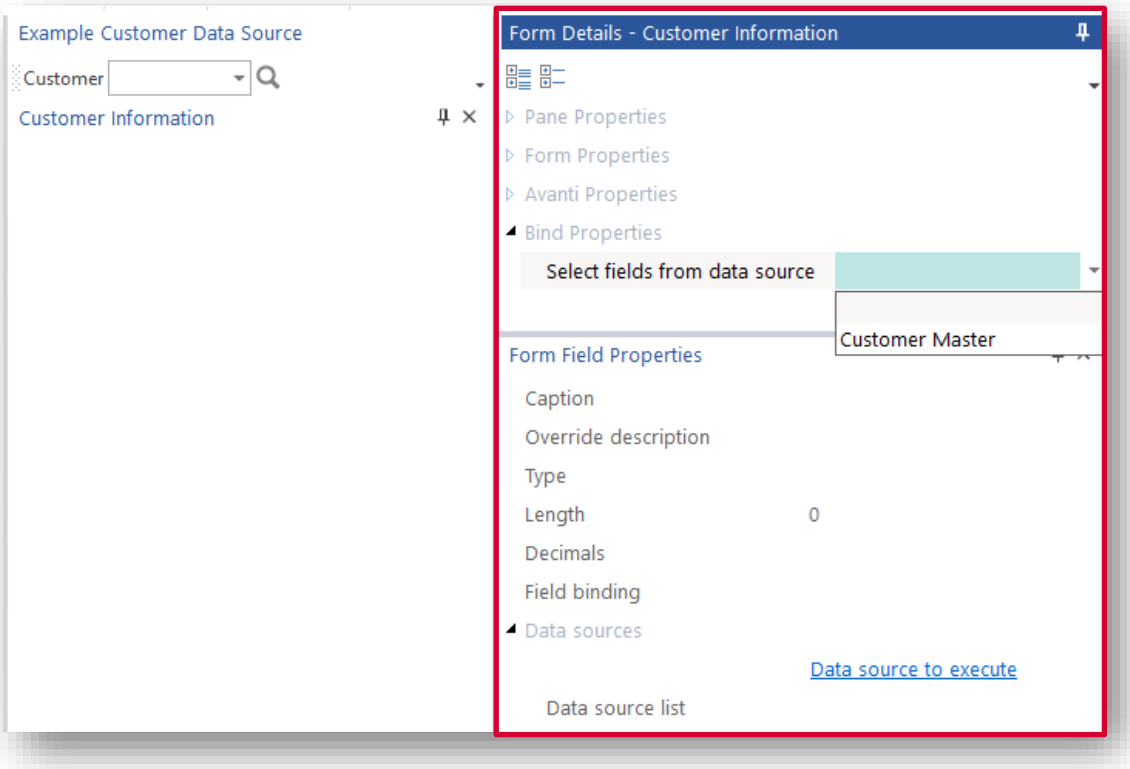
In the **Data Source Explorer** you can see where a data source will be executed in the application:



STEP 2: ADD FIELDS TO A FORM USING A DATA SOURCE

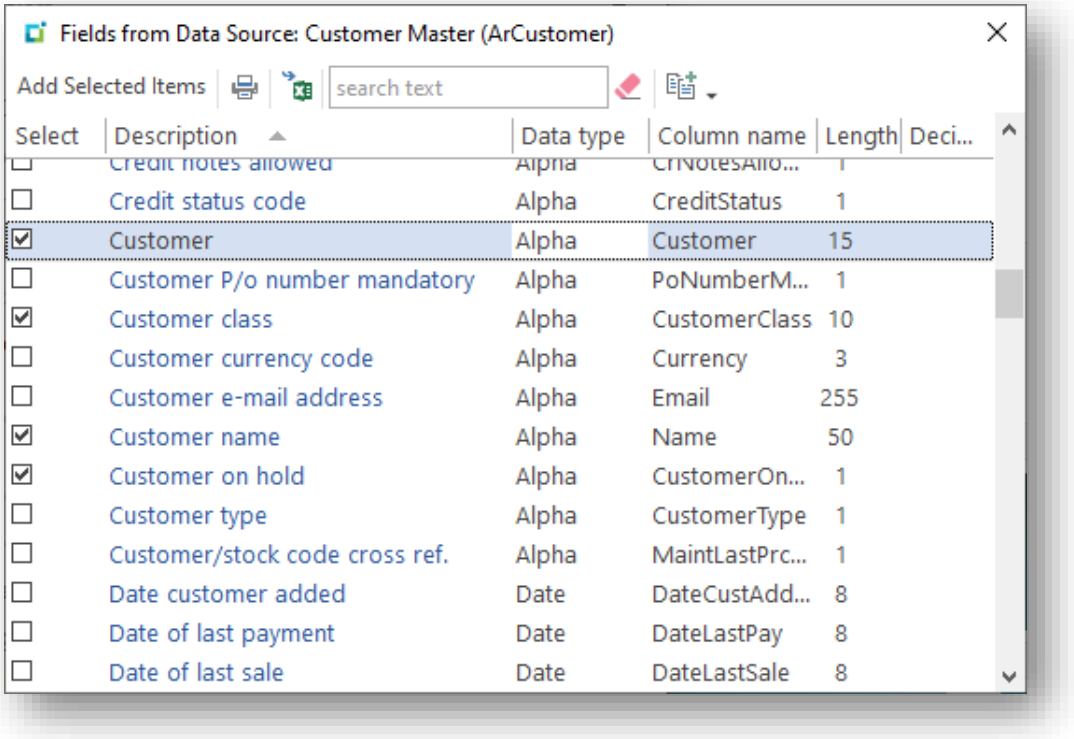
We can now add fields to the form using the data source we have just created.

Click on the Customer Information pane and you will see the **Form Details** and **Form Field Properties** panes are displayed.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

From the **Select fields from data source** field, select **Customer Master** from the dropdown list. All the fields that are derived from the `ArCustomer` master table will be shown in a list view:



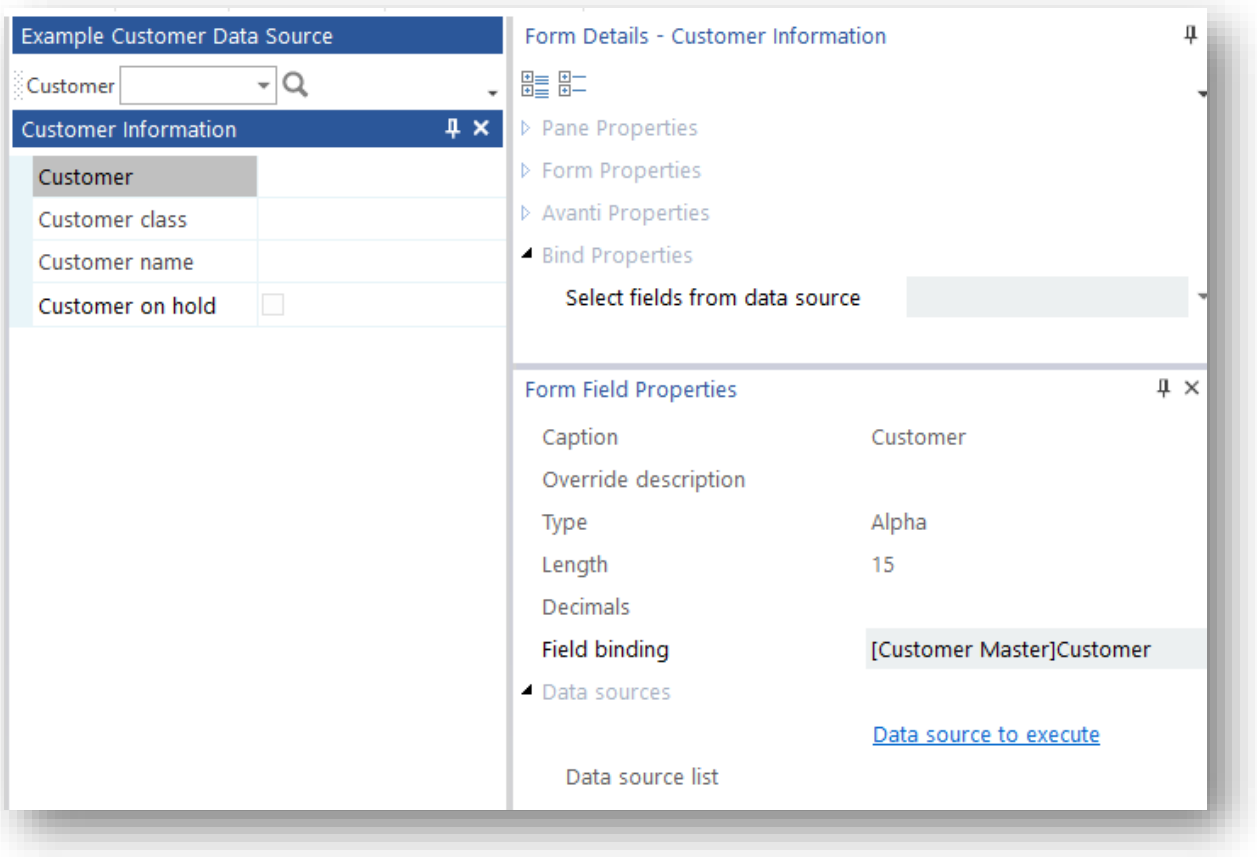
You can now select which fields you want to add to the form, in one of two ways:

- You can click on a hyperlinked description (this will immediately add the item to the form).
- Or you can check which items are to be added in the **Select** column and then click **Add Selected Items**.

In the example above, the *Customer*, *Customer class*, *Customer name* and *Customer on hold* fields have been selected.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Click on **Add Selected Items** and the form will look as follows:



As you click on a form field, so its properties are displayed.

You will notice that there is a **Field binding** property that indicates if the field is bound to a data source.

This field can be edited to change the data source name and binding field name, but currently there's no validation if this binding property is changed.

Considerations:

- The binding property can also be seen when designing a Form in the **Tag** property, although there the property is read-only.
- You can change the form field's caption or Override description from this window.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

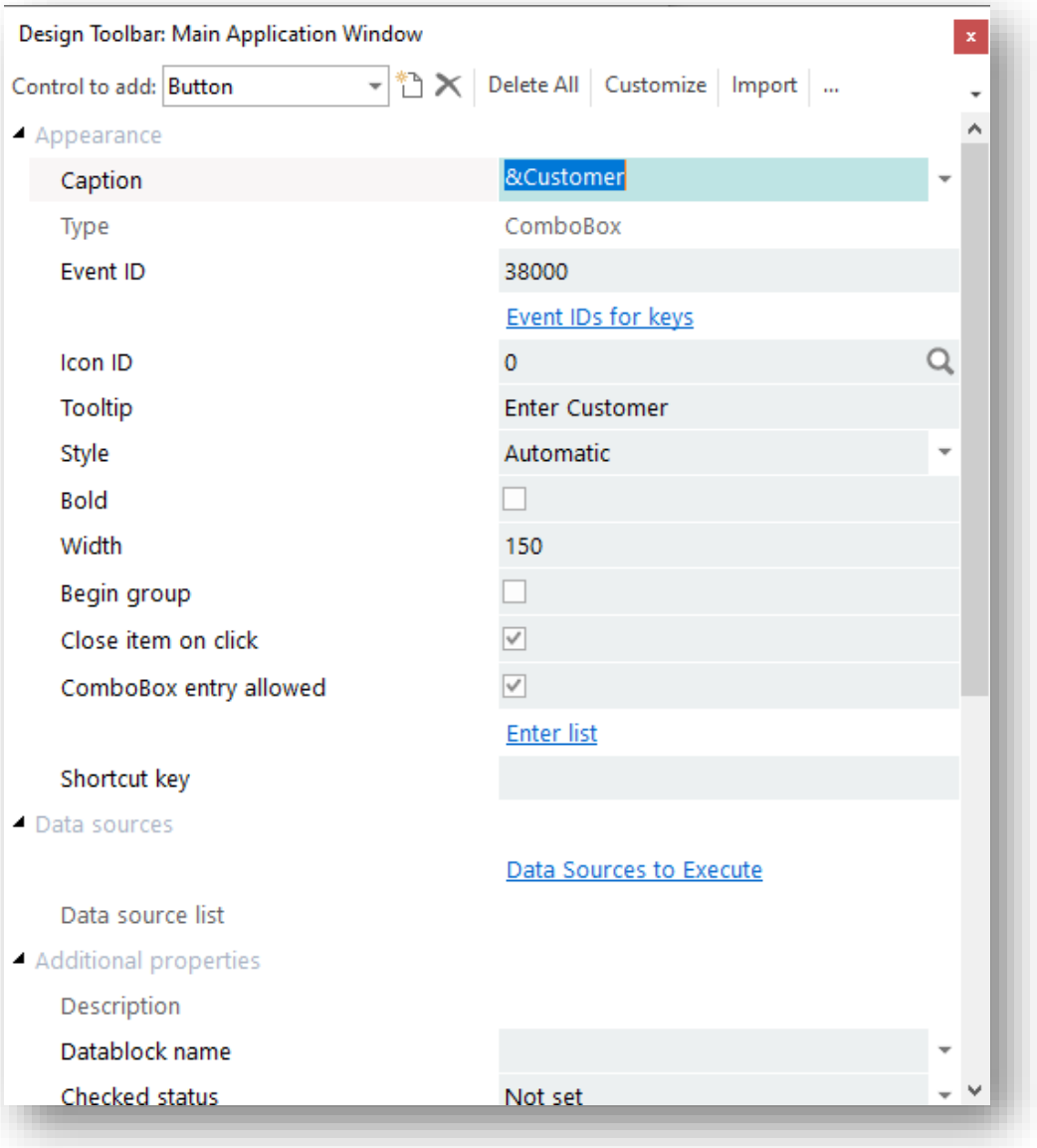
STEP 3: EXECUTE THE DATA SOURCE

At this point, the application has a form with fields bound to a data source. But there's no indication yet as to how the data source will be executed to update the form field values.

There are several ways of executing a data source:

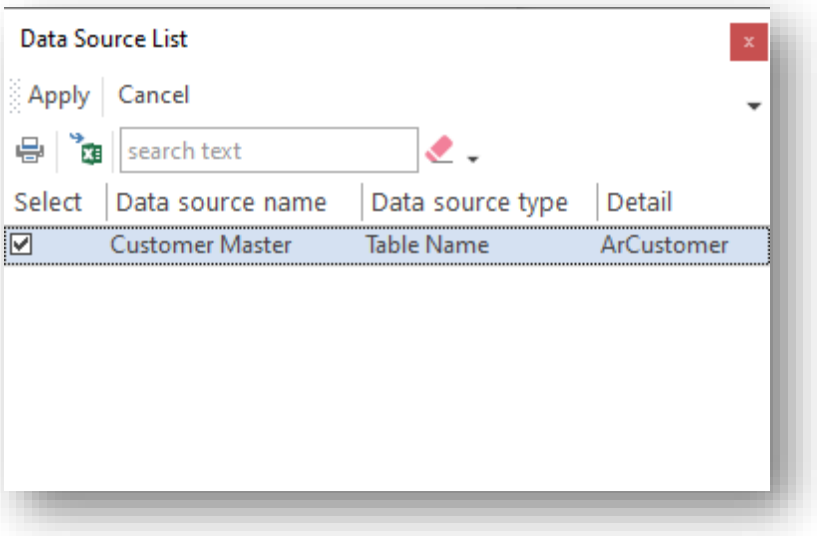
- When a toolbar value changes
- When a form field value changes
- When a list view row is 'selected'
- When a list view cell value changes
- When clicking on a chart data point
- Executing a data source from script

To execute the data source when a toolbar value changes, select to Design the toolbar, and then select the **Customer** control:



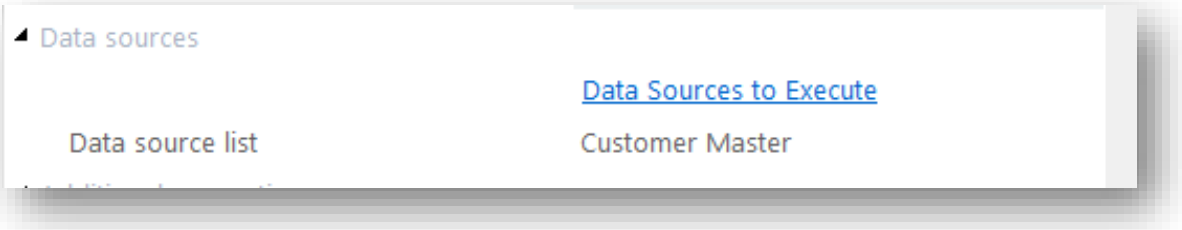
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Select the **Data Sources to Execute** hyperlink and a window is presented indicating all the data sources:



Select the **Customer Master** item and click **Apply**.

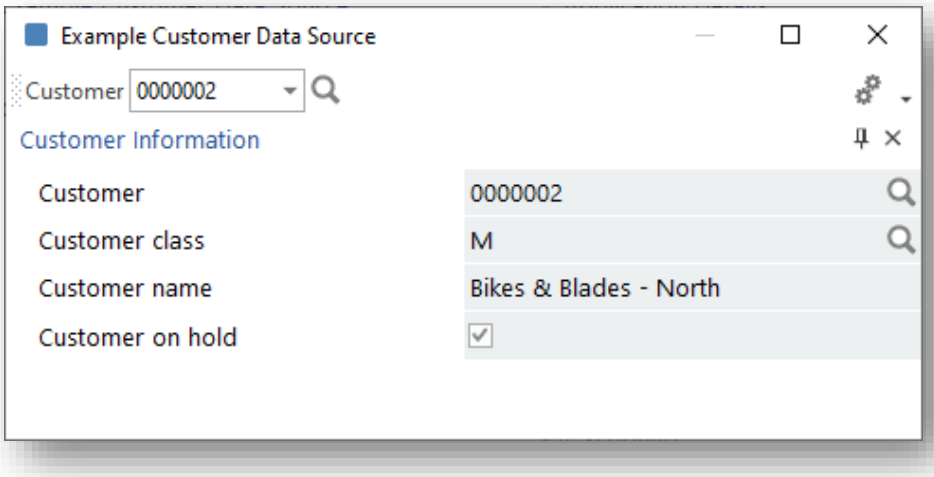
The Data Source list will now reflect this:



STEP 3: RUN THE APPLICATION

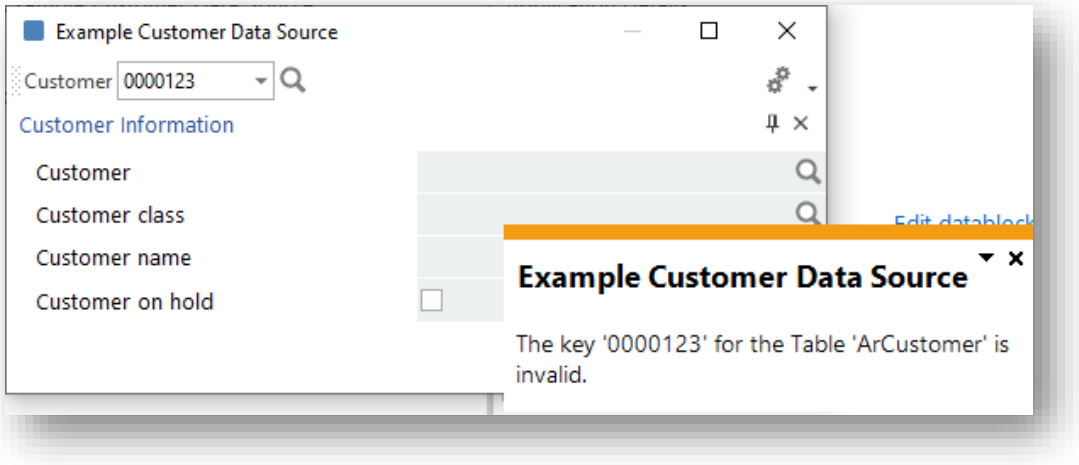
You can now run the application by pressing F5.

Enter a customer number and your form fields will be refreshed with values derived from the Customer Master table:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Now try entering a customer that doesn't exist:

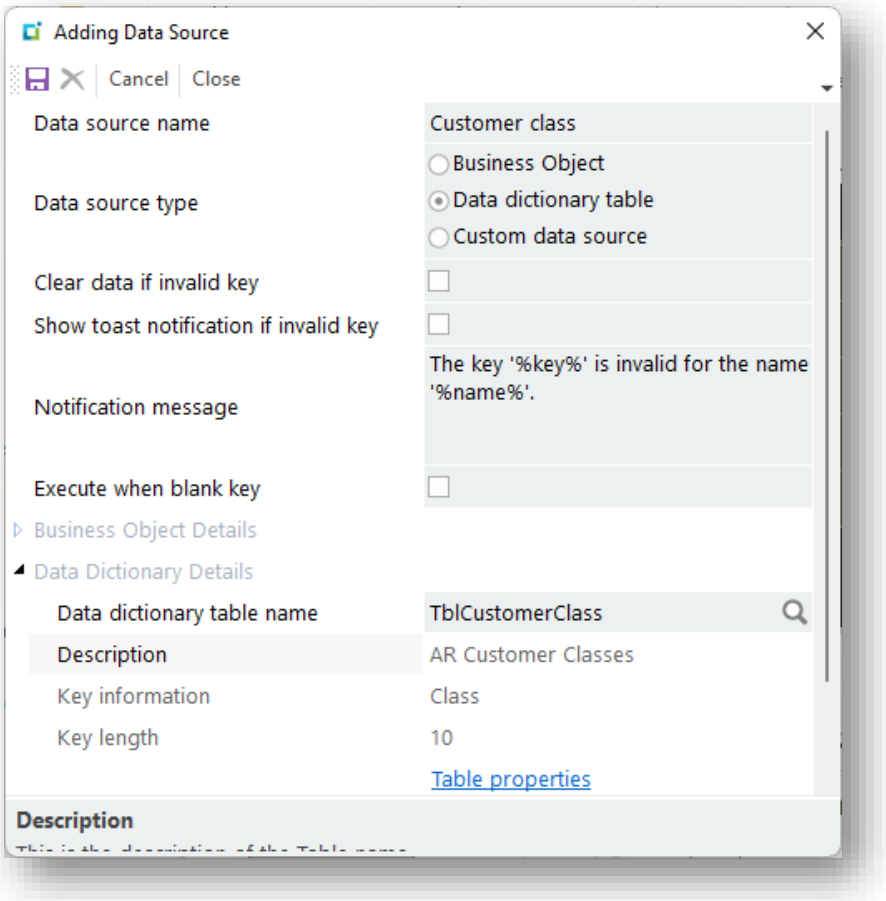


You will see a toast notification indicating the invalid key, and you will notice that the form field values have been cleared.

How to show the Description of a 'key' item

In our application we have a **Customer class** form field. It would be useful to show the Customer class description beneath this field. Here's how to do that:

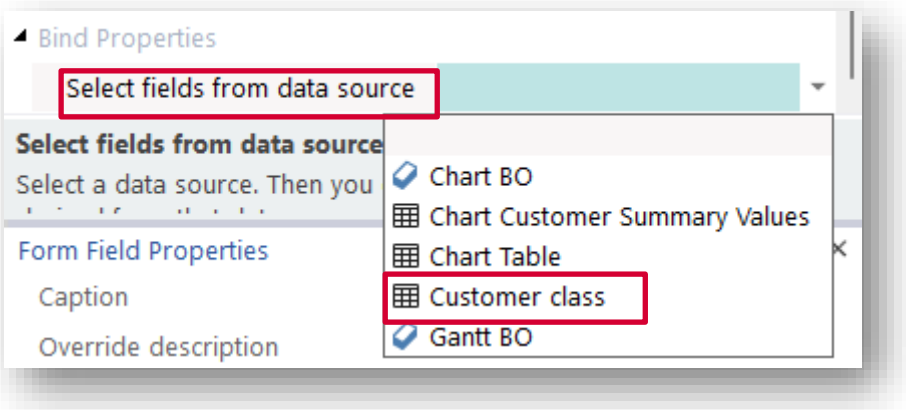
First, create another data source using the table **`TblCustomerClass`**:



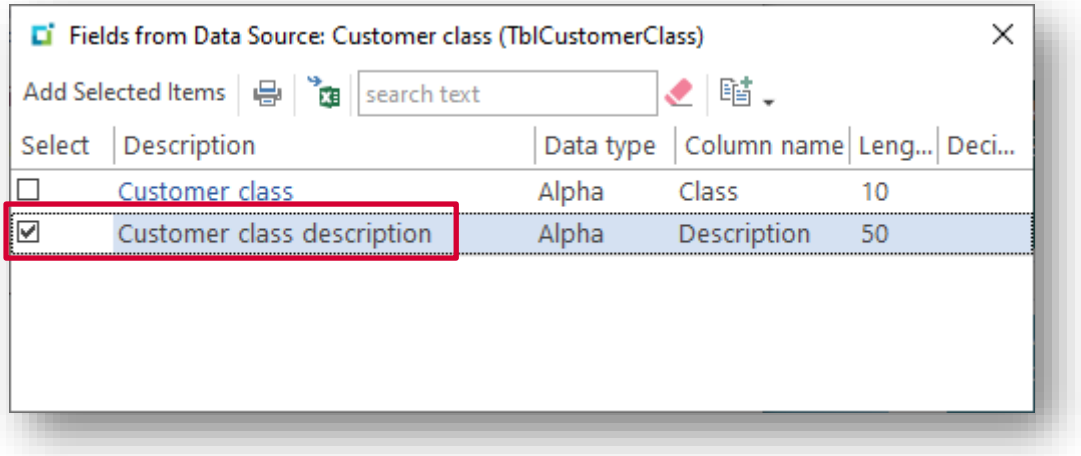
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Now we can add the Customer class description field to the form.

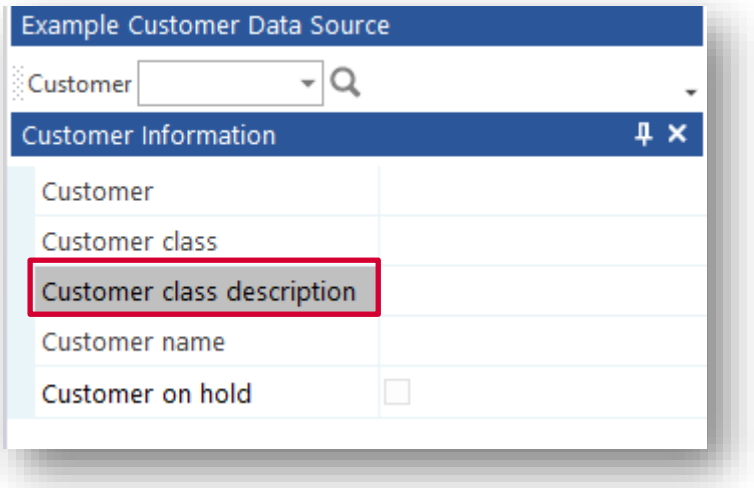
Select **Customer class** data source:



And then select the **Customer class description** field:

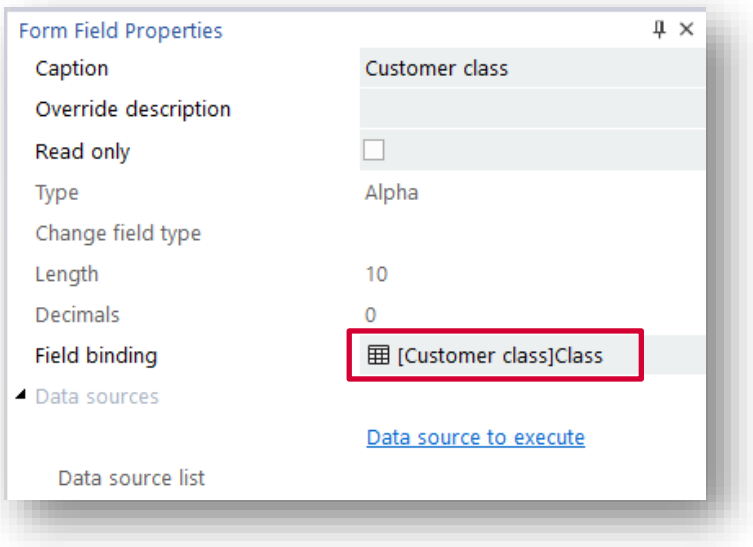


Drag the **Customer class description** to be beneath the **Customer class** field:

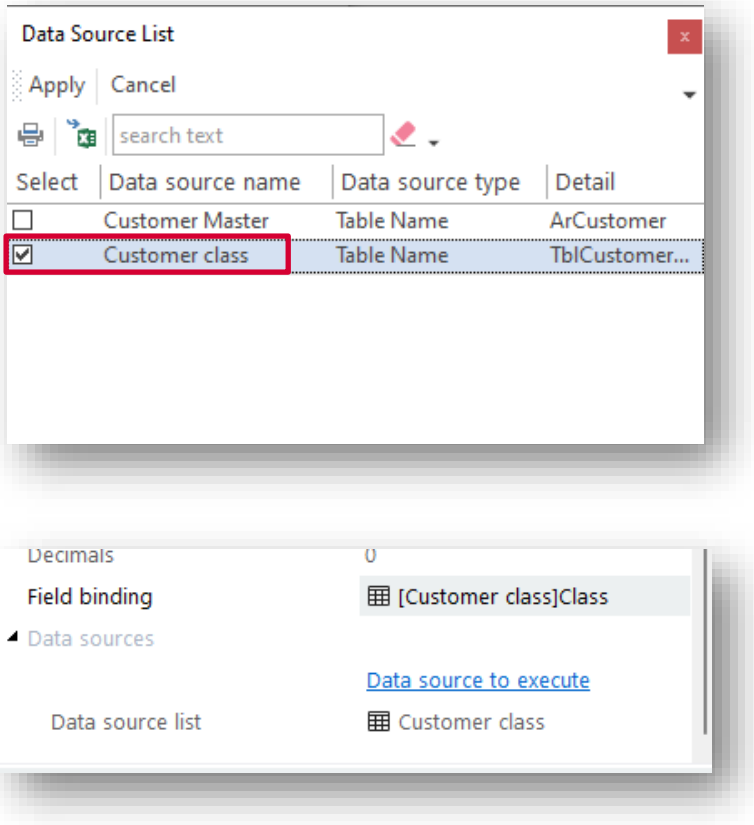


© SYSPRO. All Rights Reserved. All trademarks are recognized.

And finally, click on the **Customer class** form field and select the **Data source to execute** function:



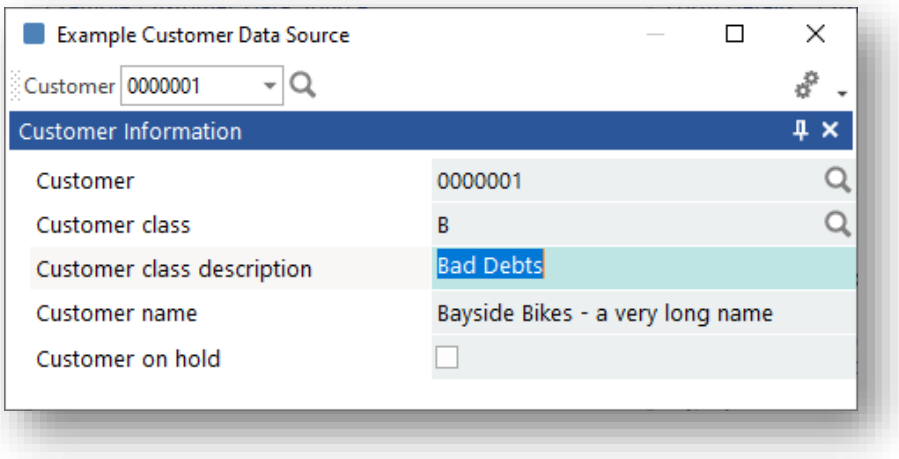
Select the **Customer class** data source and click **Apply**:



Now run the application by pressing F5.

Enter a customer number and then type a new customer class into Customer class field – you will see that the class description is now refreshed:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



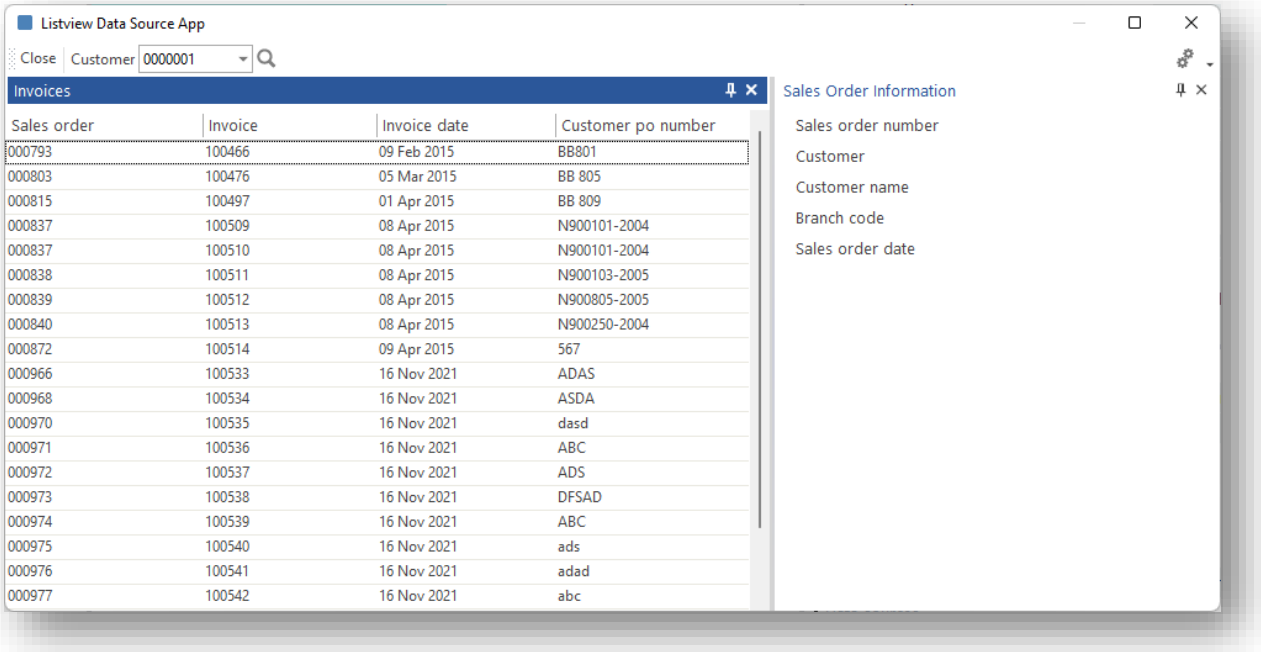
Note: When a form field value is changed (either by the application's script or because a data source was executed which in turn changed a form field value) then any data source associated with that field will be executed.

Executing a Data Source for a List view

Row 'selected' event

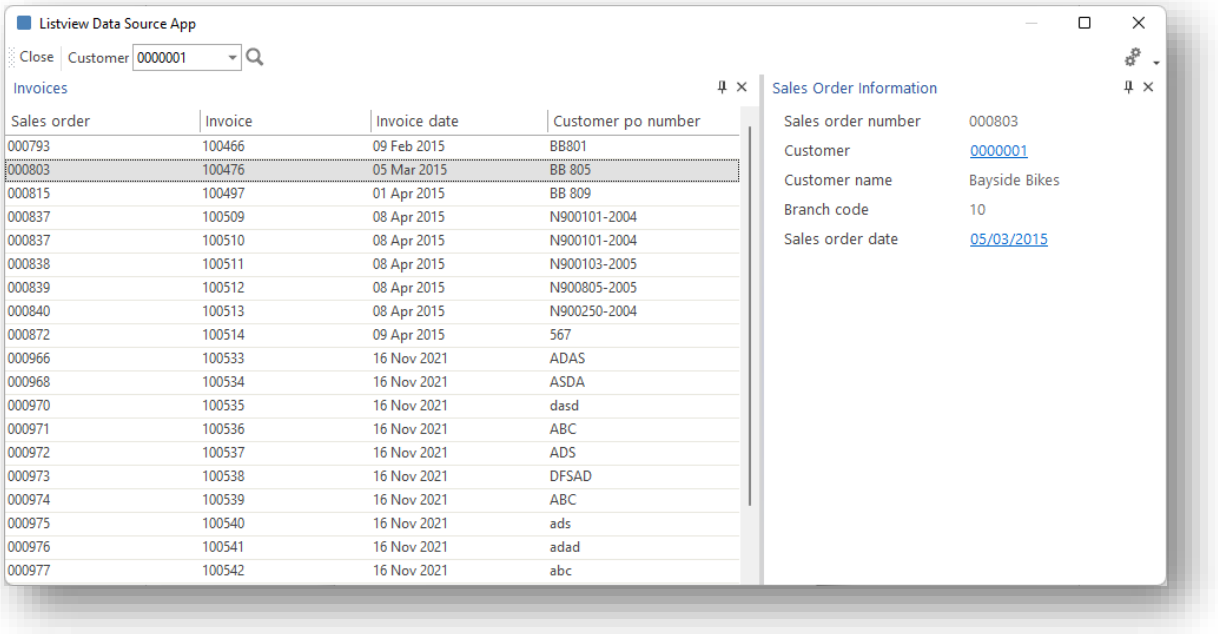
When a row becomes 'selected' (i.e. becomes the focused row) you can execute a data source using the value of a specific column to be applied to that data source as the KEY. This might be useful if you wish to show some more information in a separate Form using a data source.

The following is an example application that shows a list of INVOICES for a Customer:



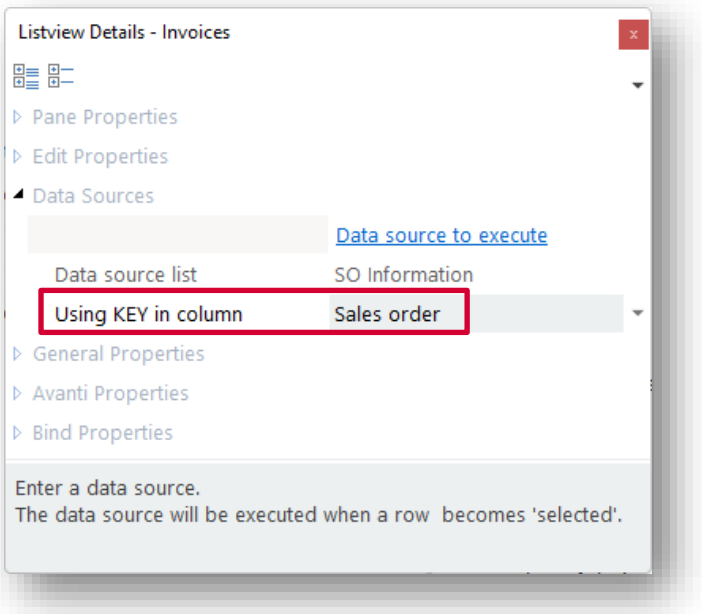
© SYSPRO. All Rights Reserved. All trademarks are recognized.

When you click on a row, the information about the Sales Order is then shown in a separate form. In this example, the form fields in the form **Sales Order Information** are bound to the data source **SO Information**.



To achieve this using a data source:

1. Against the List view properties pane, select the appropriate **Data Source to Execute**. In this example, a data source of **SO Information** has been chosen (this points to the table `SorMaster`).
2. Select the name of the column in which the KEY to execute the data source is found, using the property **Using KEY in column**.
   In this example, we will use the SALES ORDER column:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

And now, when a row is selected, the data source SO Information is executed using the KEY found in the column Sales Order, which in turn will populate the Sales Order Information form.

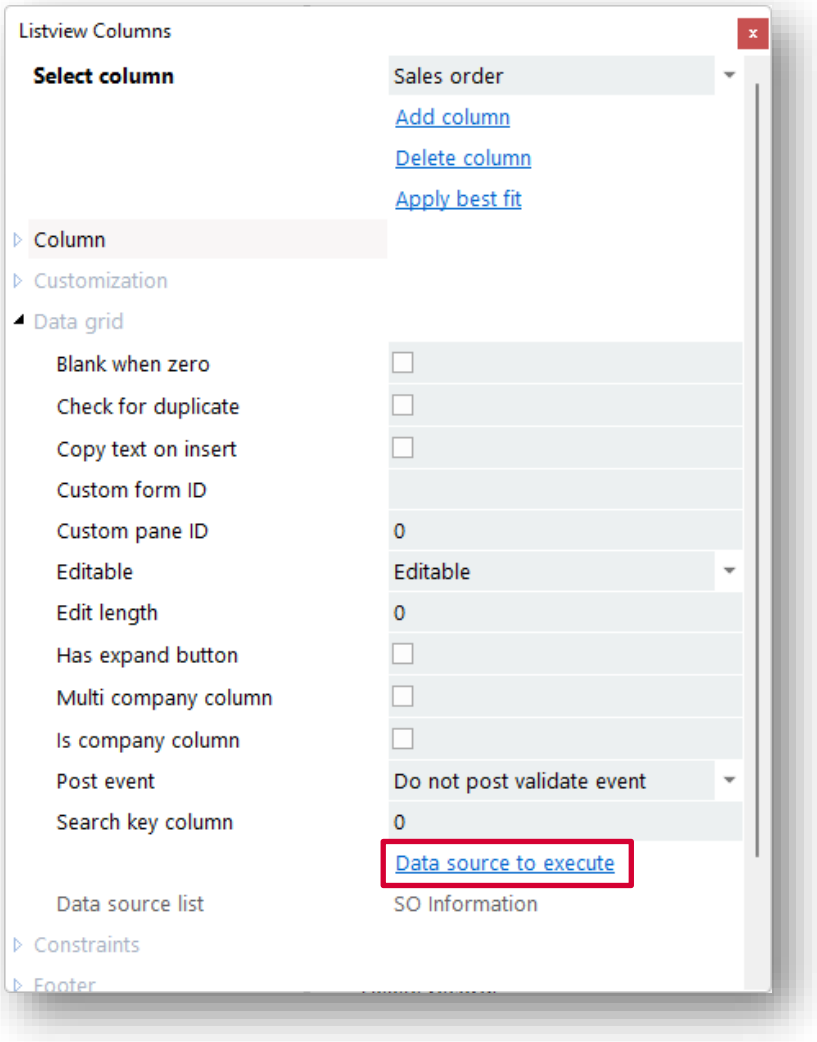
CELL VALUE CHANGED EVENT

When a cell in a list view is changed, you can execute a data source using the value of the cell to be applied to that data source as the KEY.

Typically, you would use this to validate the cell value. If the cell value is not a valid KEY then a toast notification is displayed, and focus is set back on the cell. If the cell value is a valid KEY, then the data source field bindings are executed.

To set this up, proceed as follows:

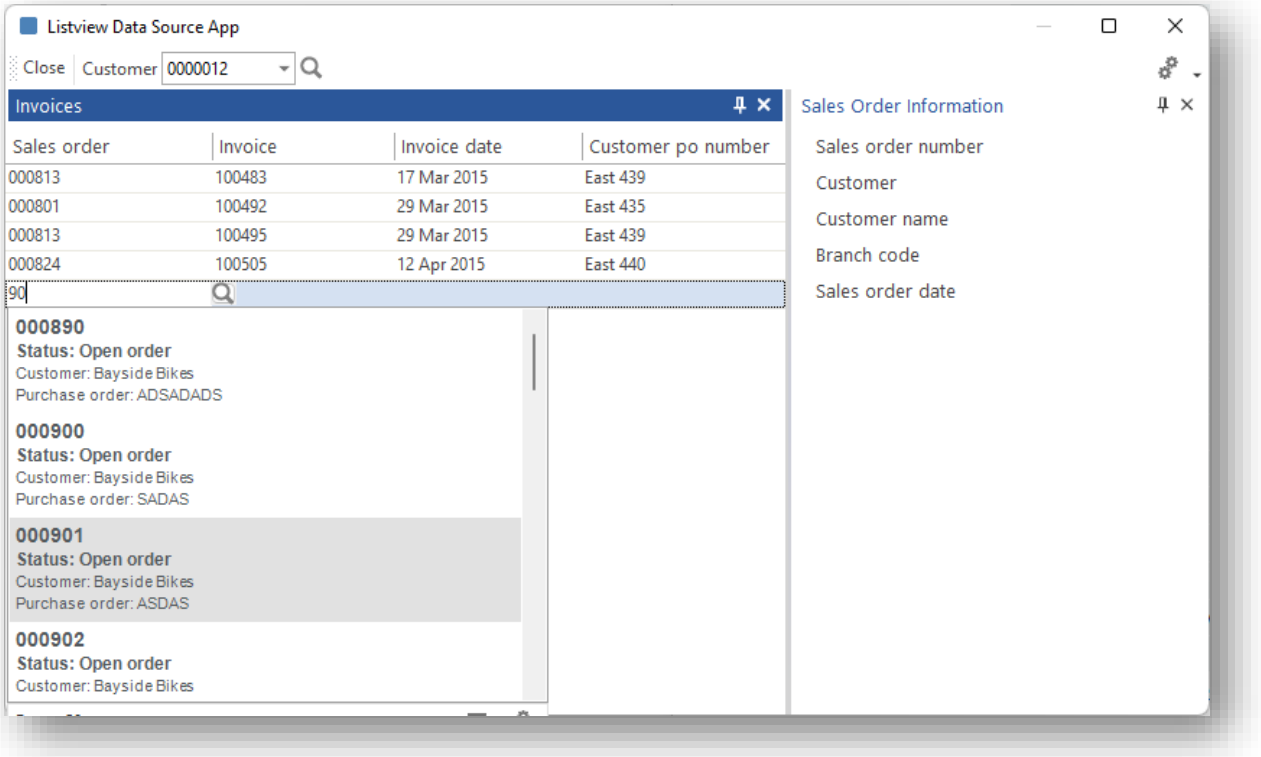
1. In the Listview Columns window, select the **Data Source to Execute** hyperlink and the required data source:

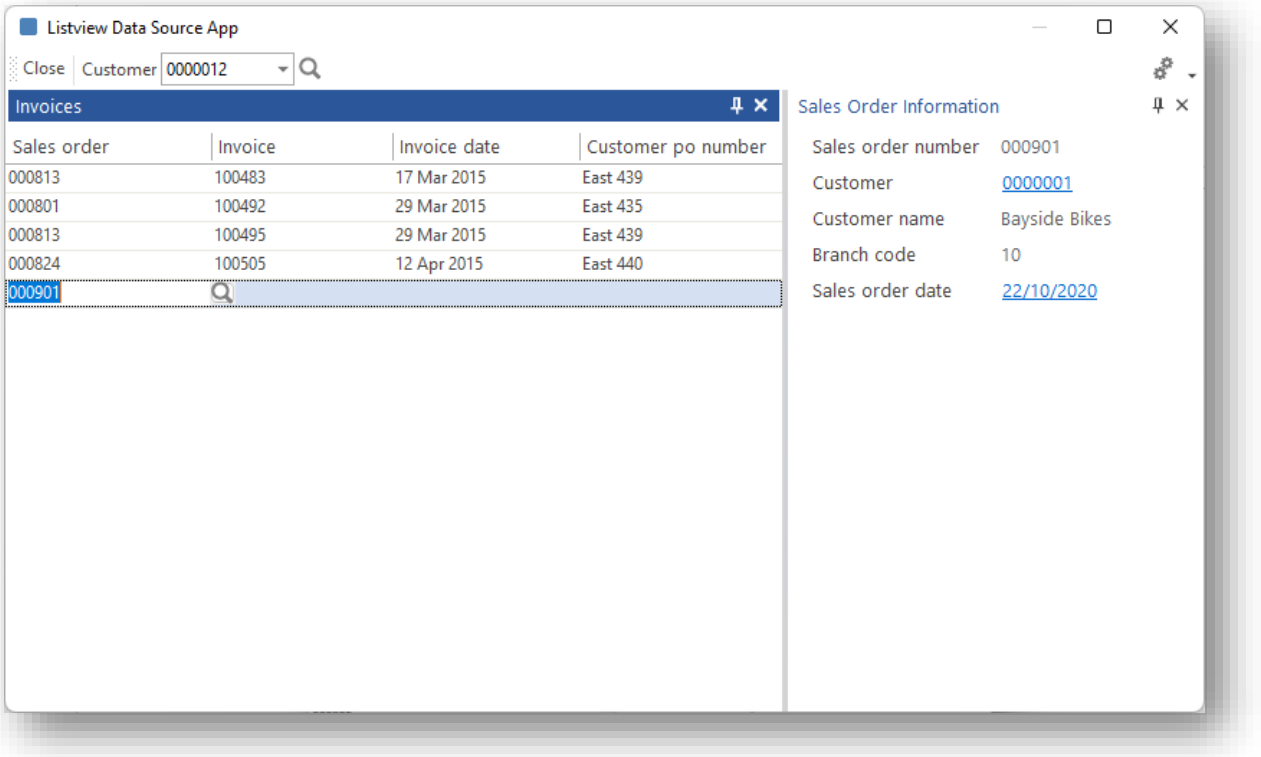


Note: The column must be defined as **Editable** for this to work.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

So, using our example, enter a Sales Order in the cell and the Sales Order Information form will be populated:





© SYSPRO. All Rights Reserved. All trademarks are recognized.

EXECUTING A DATA SOURCE VIA CODE

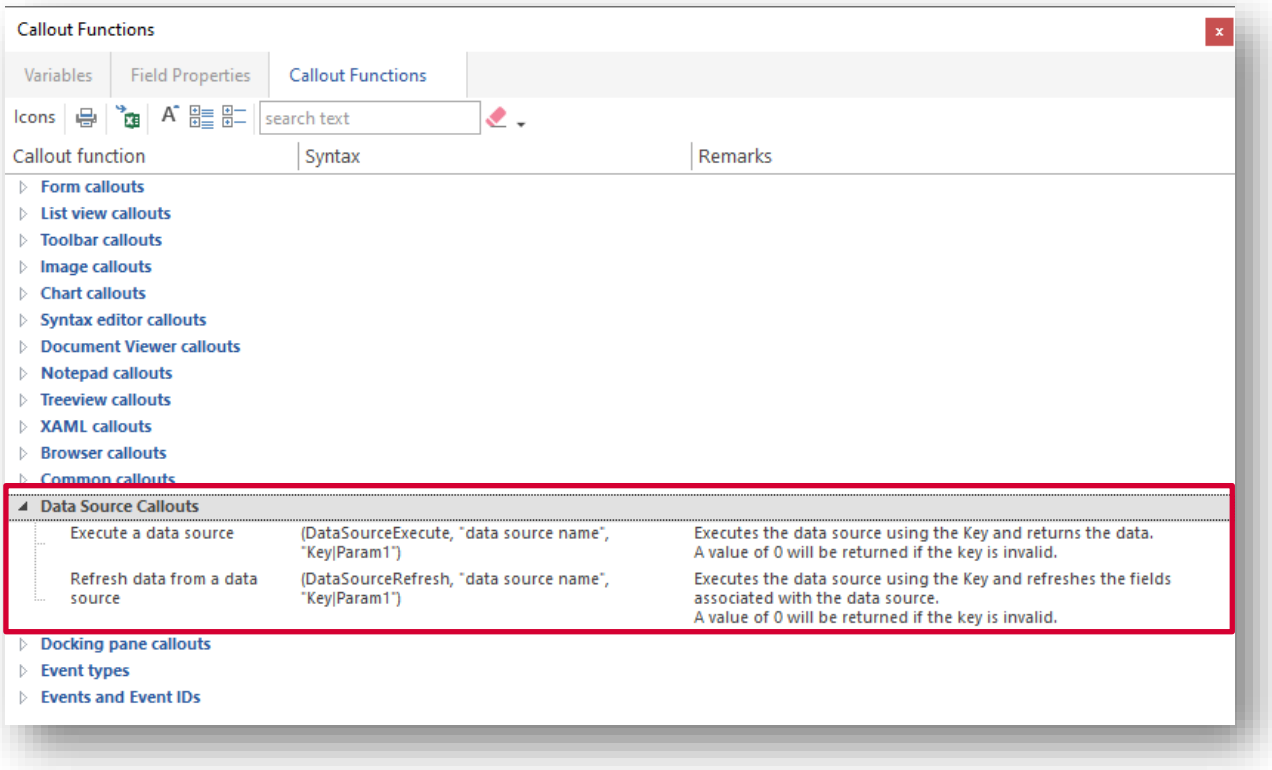
You can also execute a data source via script.

This may be very useful where the application needs to refresh form field data at some point in the logic, rather than when a toolbar or field value changes. Additionally, the data source in question may require a compound 'key' to be passed to it – for example, a Stock code AND a Warehouse value.

In these cases, you can use two data source callouts, as indicated in the image below:

- `DataSourceExecute` callout:
  This callout executes the named data source with a KEY and returns the data from that data source.
- `DataSourceRefresh` callout:
  This executes the name data source with a KEY and automatically refreshes any form fields bound to that data source.

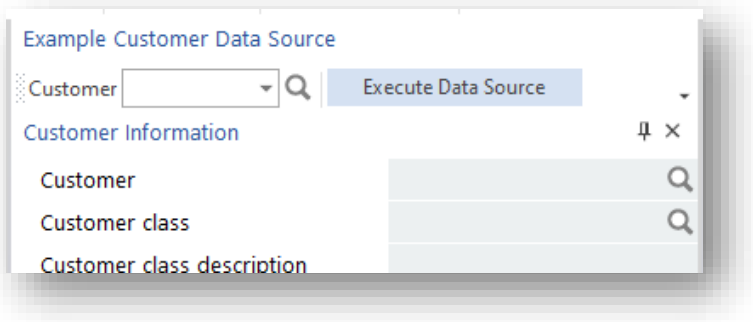
In both cases, if the KEY being passed is invalid then the **ReturnValue** is **0** .



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Let's see this working:

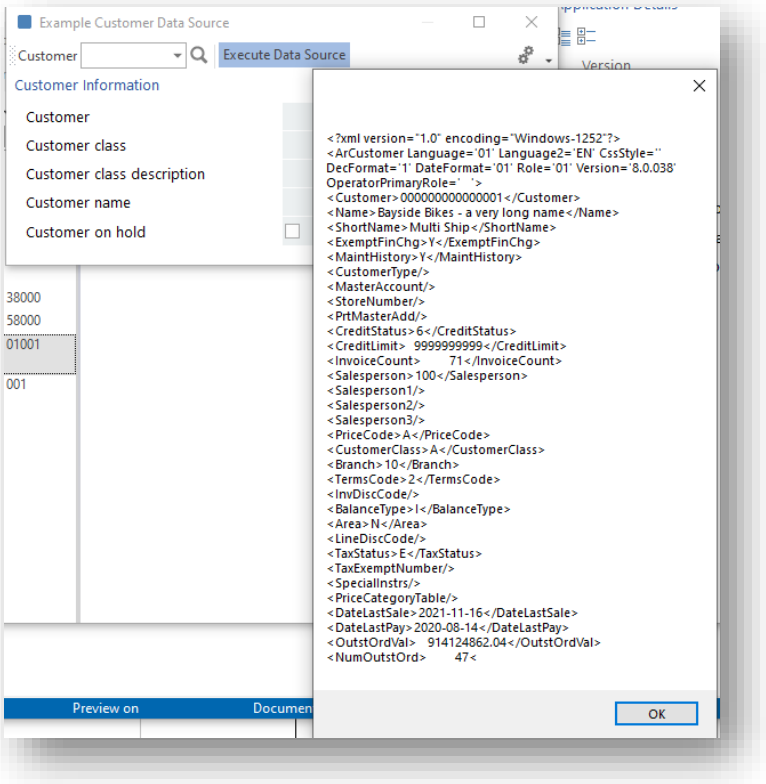
In our application, we can add a button labelled **Execute Data Source**, whose control ID is *1001*.



Next, in the script, we can add this code:

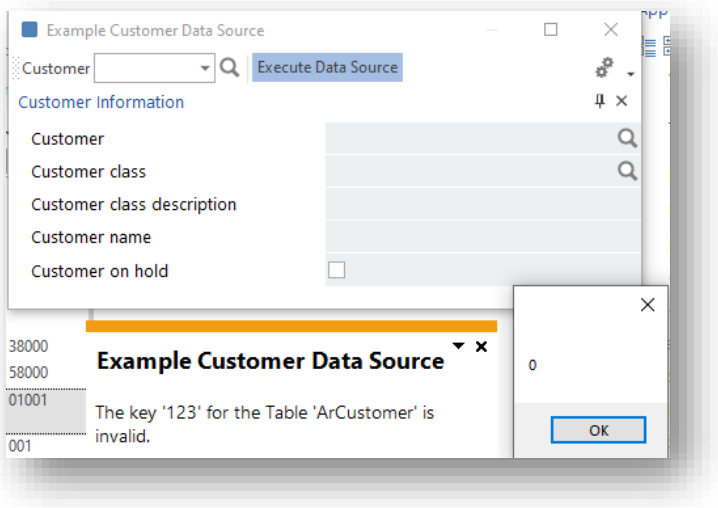
```
Function AppDesigner_OnUserEvent(EventType,EventName,EventID,Param1,Param2,Param3)

if EventType = ToolbarEvent then
   if EventID = 1001 then
      returnValue = CallSYSPROFunction(DataSourceExecute, "Customer Master",
"1|Param1")
      msgbox ReturnValue
   end if
end if

End Function
```

When clicking on the button, the data source **Customer Master** will be executed with a KEY being passed of '1' and a message box will be displayed with the contents of the data returned from the data source:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

If we change the script to pass an invalid key, this is what happens:



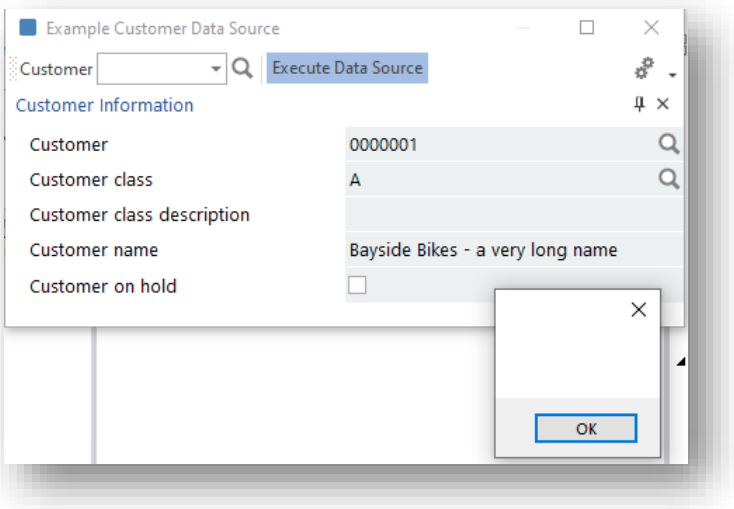
A toast notification is given indicating the invalid key, and the return value from the data source is '0'.

It's at this point you may wish to change the **Show toast notification if invalid key** option against the data source to be unchecked – this is useful if the script is to decide what action to take for an invalid key instead.

Now let's change the callout to be an automatic refresh of form fields:

```
if EventType = ToolbarEvent then
   if EventID = 1001 then
      returnValue = CallSYSPROFunction(DataSourceRefresh, "Customer Master",
"1|Param1")
   end if
end if
End Function
```

And when we click on the button, this is what happens:



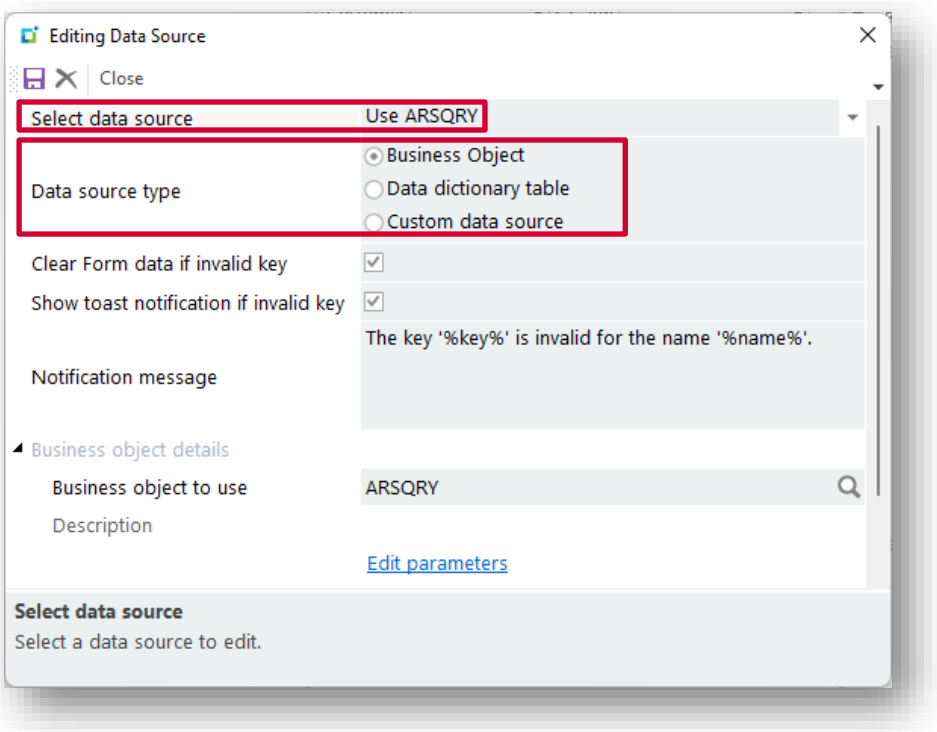
The form fields are now refreshed, and the return value is blank indicating the callout was successful.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Using a Business Object as a Data Source

A SYSPRO business object provides more information that can be used as a data source than a Table.

In the following example, a data source name of **Use ARSQRY** has been defined and the **Business Object** option has been selected for the Data Source type.

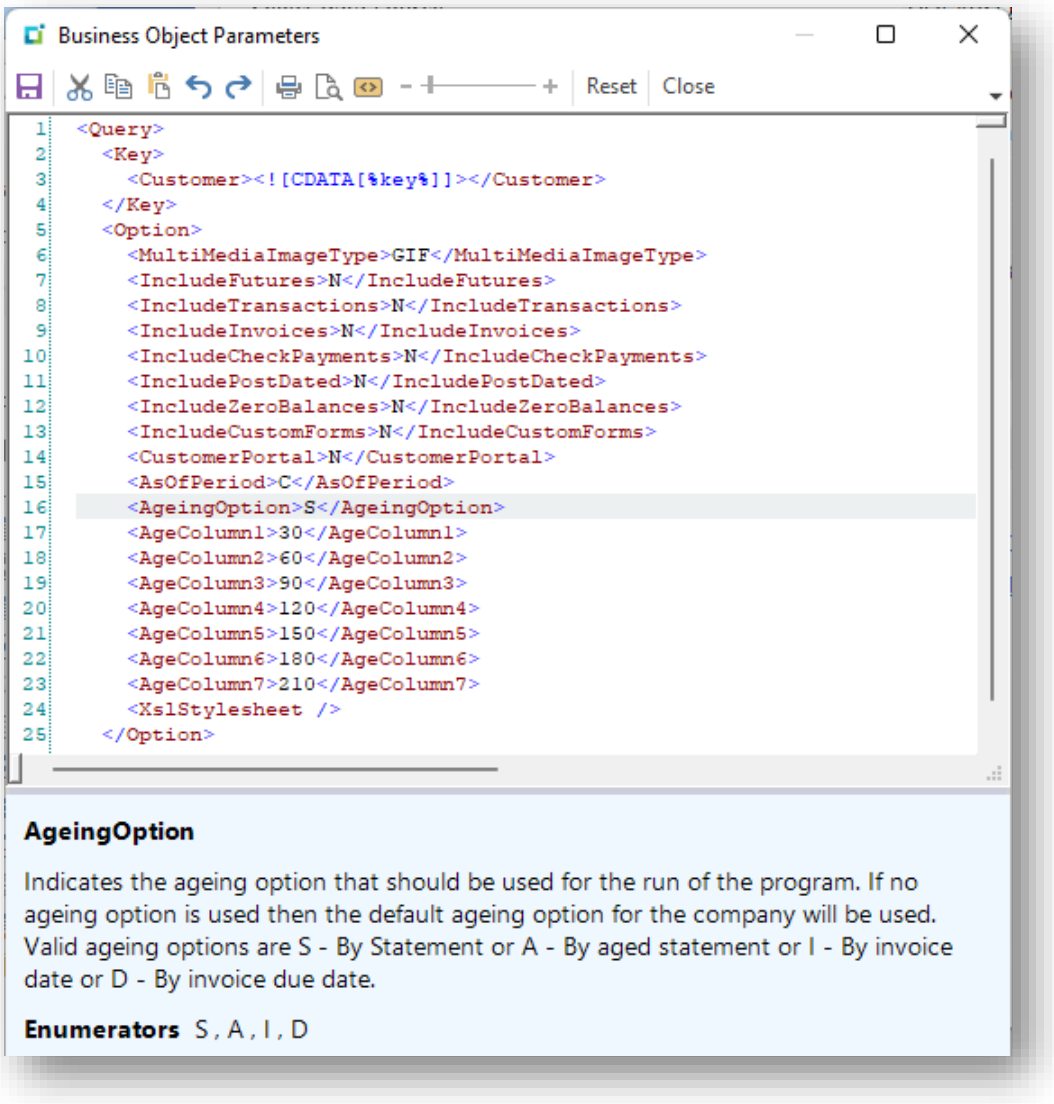


To use a SYSPRO Business Object as a data source, proceed as follows:

1. Supply the name of the business object.
   Considerations:
   - This must be of type Query.
   - After entering a business object, a default set of input parameters is created based on the XML derived from the `…\base\Schemas` folder using the business object name (e.g. `…\base\schemas\ARSQRY.XML`).
2. Edit the parameters to be supplied for the business object by clicking on **Edit parameters**.
   An editor will be displayed with the default XMLIN parameters, including the placeholder `%key%` for the key to be supplied.

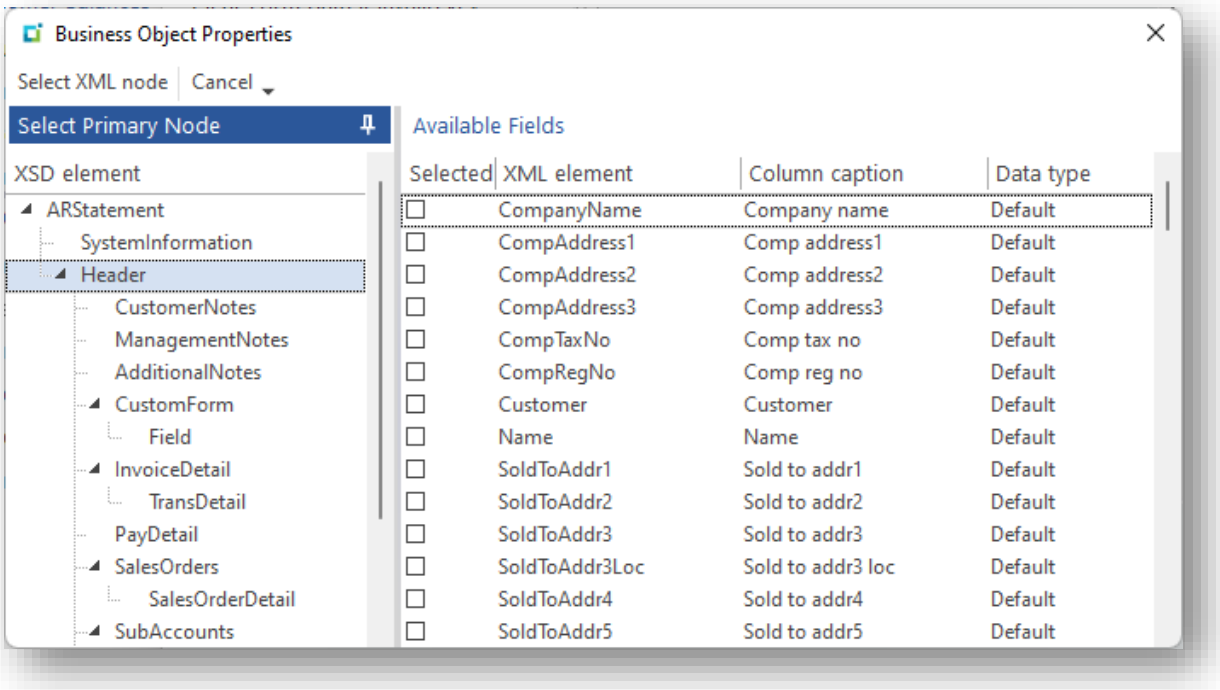
© SYSPRO. All Rights Reserved. All trademarks are recognized.

3. Click the SAVE button to save any changed parameters or click **Reset** to reset the XMLIN back to the original values.

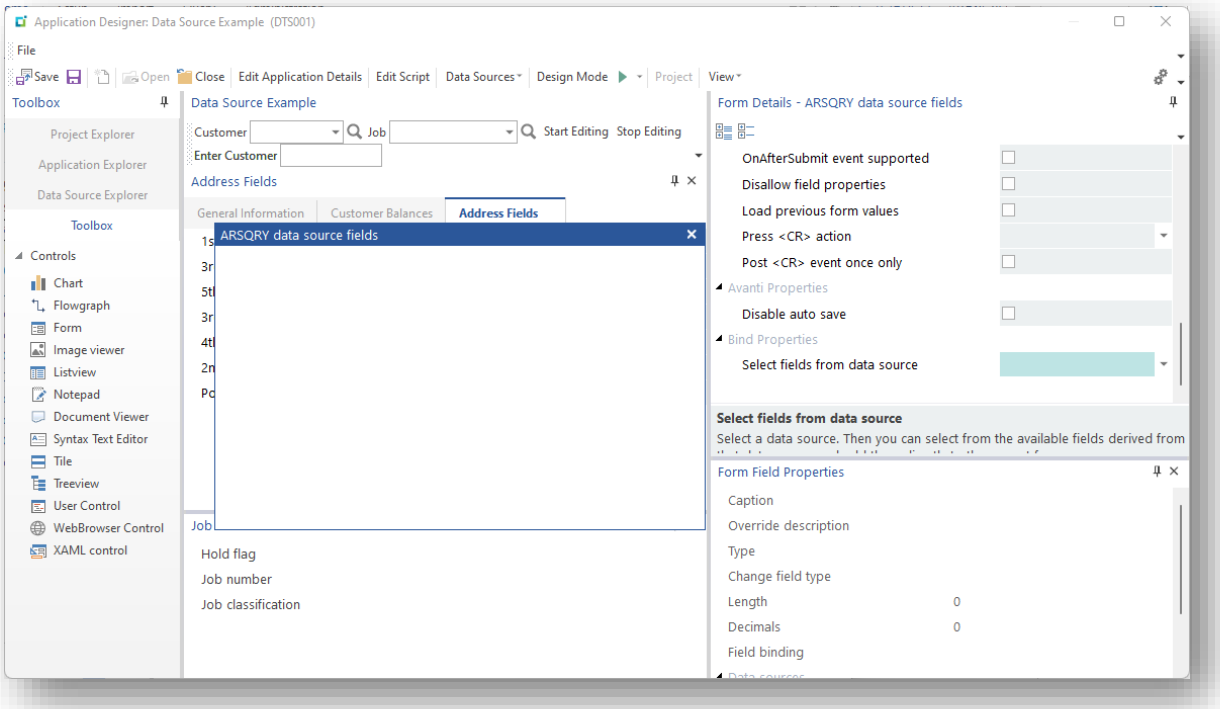


4. Select a single XML node in the data returned from the Business Object by clicking on **Select XML Node**.
   When you select a node, the available fields will be shown – these are the fields that can be selected and added to a form.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

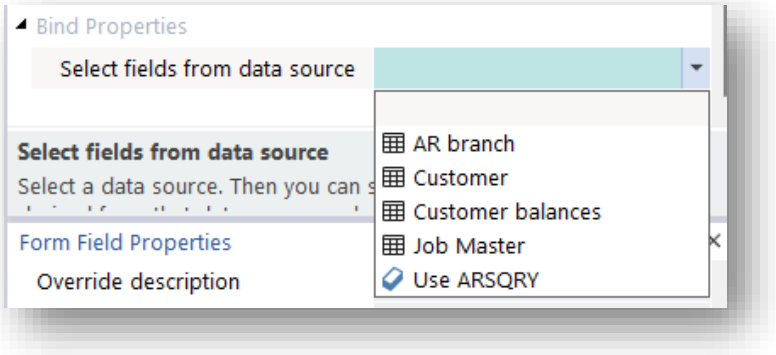


5. Once the data source has been saved, you can select the fields to be shown on a form. Here, a blank form has been created:

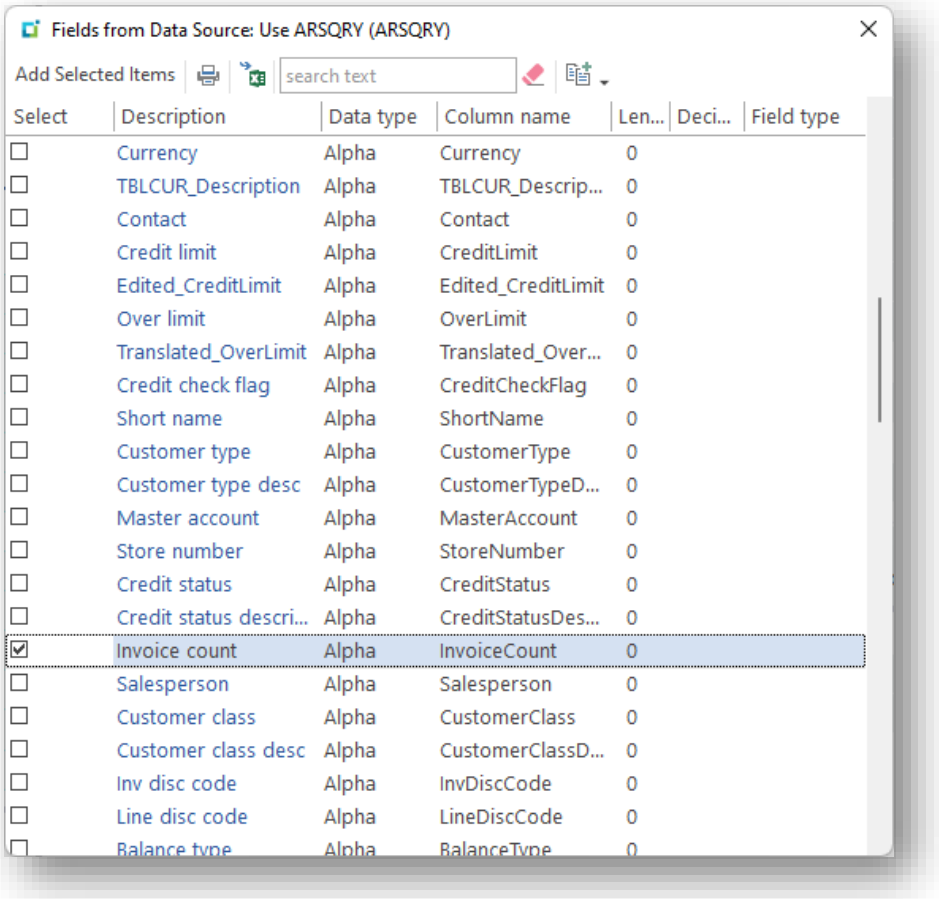


© SYSPRO. All Rights Reserved. All trademarks are recognized.

6. From the **Select fields from data source** list, select **Use ARSQRY**:



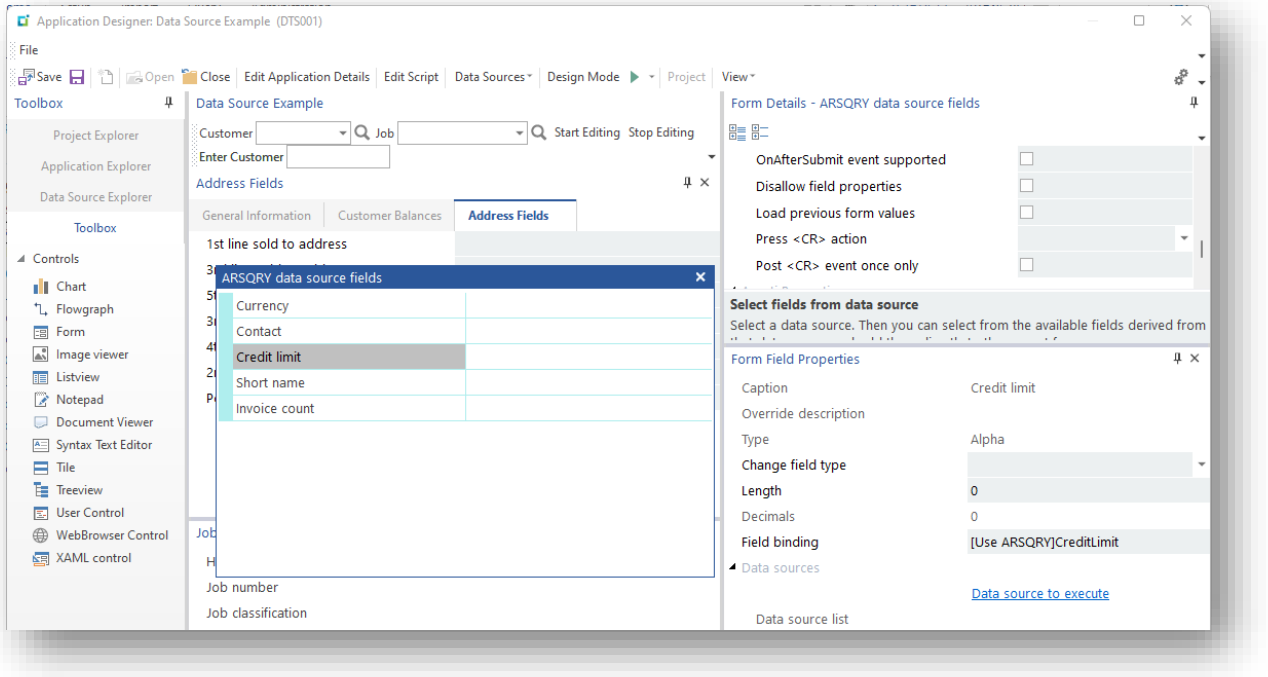
The fields derived from the **Header** XML node in the business object ARSQRY are displayed:



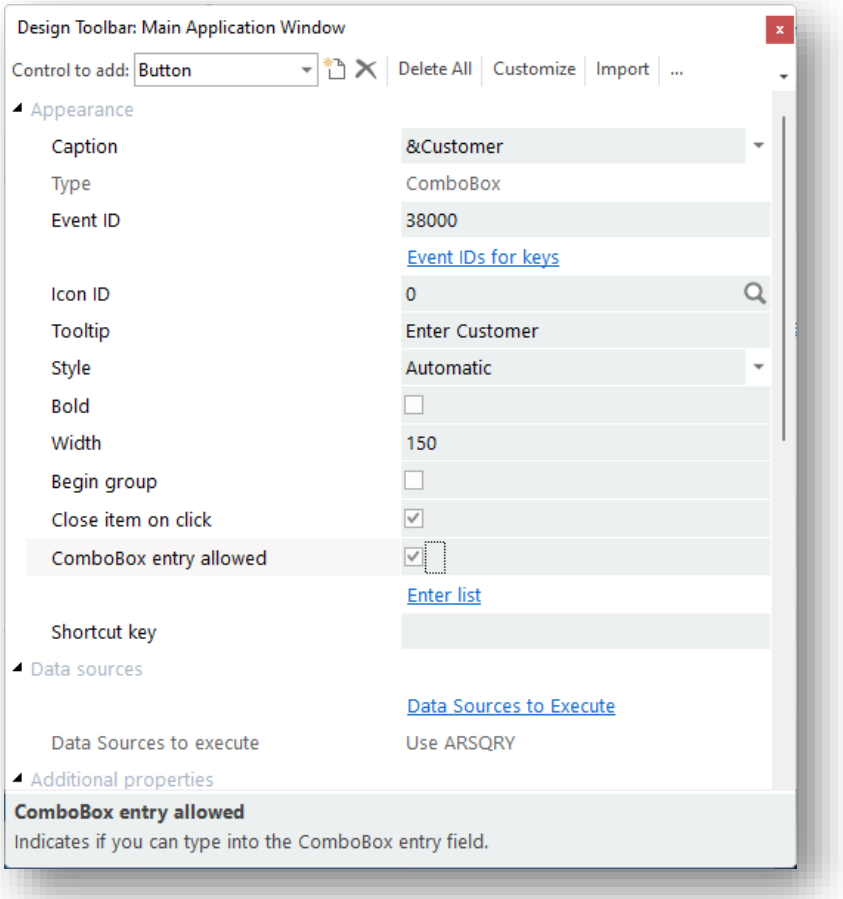
Select	Description	Data type	Column name	Len...	Deci...	Field type
☐	Currency	Alpha	Currency	0		
☐	TBLCUR_Description	Alpha	TBLCUR_Descrip...	0		
☐	Contact	Alpha	Contact	0		
☐	Credit limit	Alpha	CreditLimit	0		
☐	Edited_CreditLimit	Alpha	Edited_CreditLimit	0		
☐	Over limit	Alpha	OverLimit	0		
☐	Translated_OverLimit	Alpha	Translated_Over...	0		
☐	Credit check flag	Alpha	CreditCheckFlag	0		
☐	Short name	Alpha	ShortName	0		
☐	Customer type	Alpha	CustomerType	0		
☐	Customer type desc	Alpha	CustomerTypeD...	0		
☐	Master account	Alpha	MasterAccount	0		
☐	Store number	Alpha	StoreNumber	0		
☐	Credit status	Alpha	CreditStatus	0		
☐	Credit status descri...	Alpha	CreditStatusDes...	0		
☑	Invoice count	Alpha	InvoiceCount	0		
☐	Salesperson	Alpha	Salesperson	0		
☐	Customer class	Alpha	CustomerClass	0		
☐	Customer class desc	Alpha	CustomerClassD...	0		
☐	Inv disc code	Alpha	InvDiscCode	0		
☐	Line disc code	Alpha	LineDiscCode	0		
☐	Balance type	Alpha	BalanceType	0		

© SYSPRO. All Rights Reserved. All trademarks are recognized.

7. Select the fields to be included in the form and click **Add Selected Items**. The selected fields will be displayed on the form:



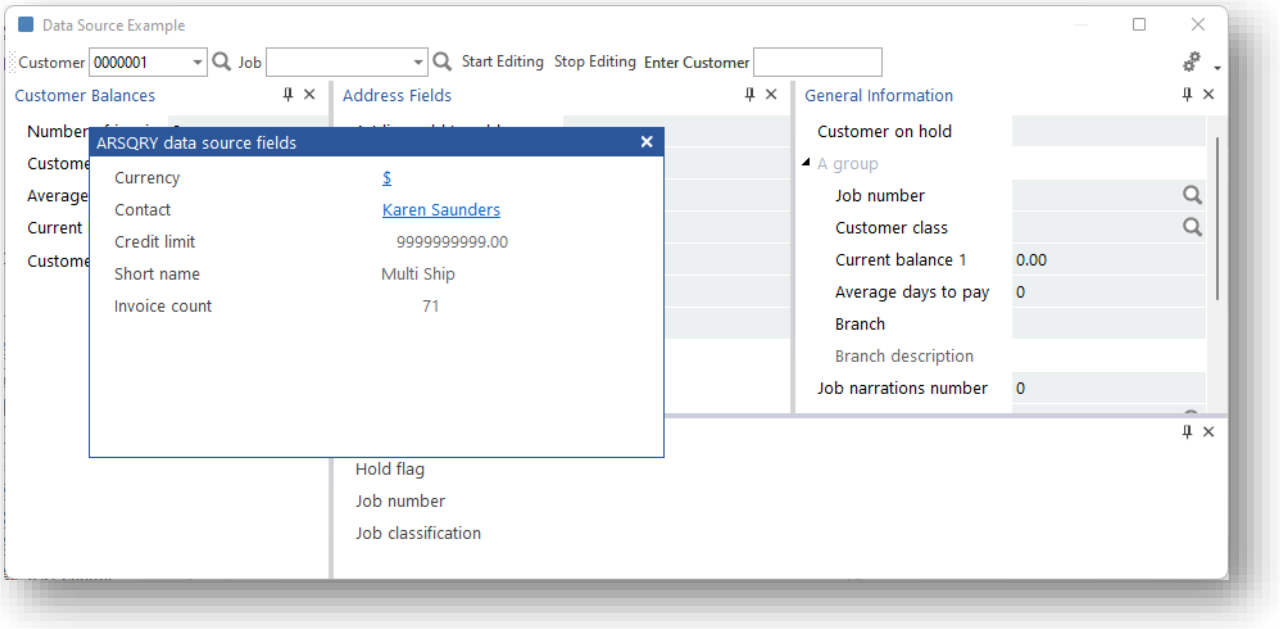
8. Finally, against a toolbar labelled **Customer** we can select the data source **Use ARSQRY** to be executed:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

You can now run the application.

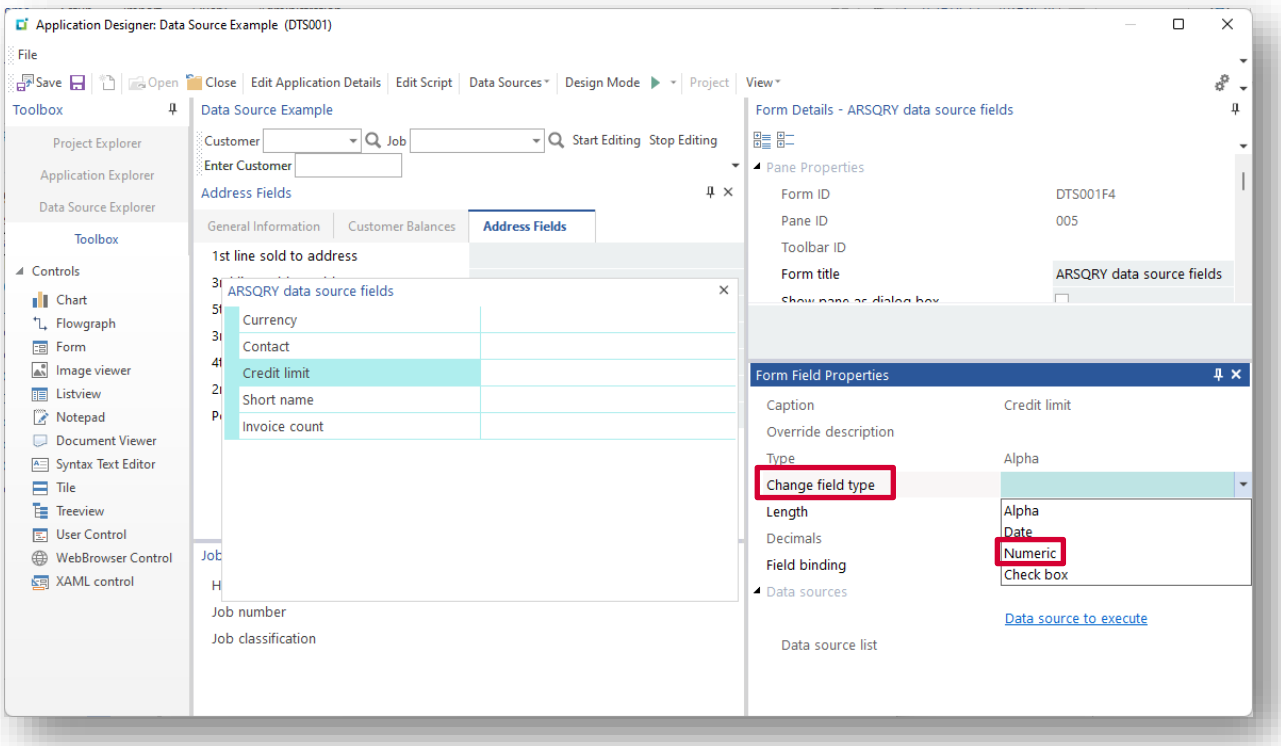
After entering a valid Customer number, you will see the form fields populated with data. However, it may be noticed that all fields have been treated as type **Alpha**:



This is because the fields derived from a business object are not type-classed.

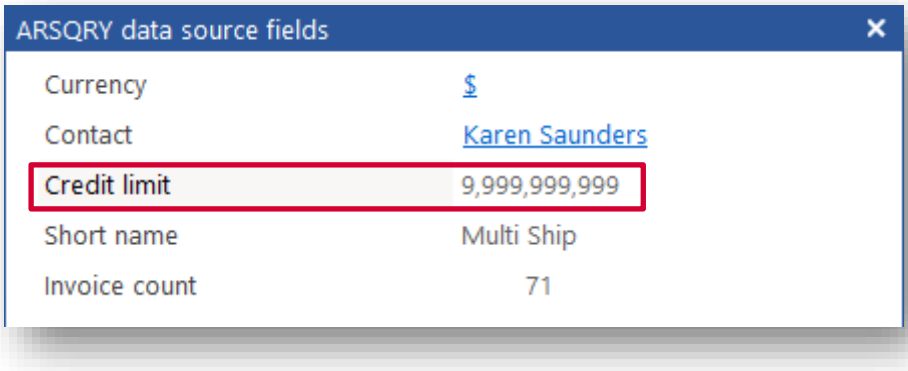
You can change the field type in the Designer:

1. Click on **Credit limit** and the Field Properties will be updated.
2. Select **Numeric** from the **Change field type** list:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

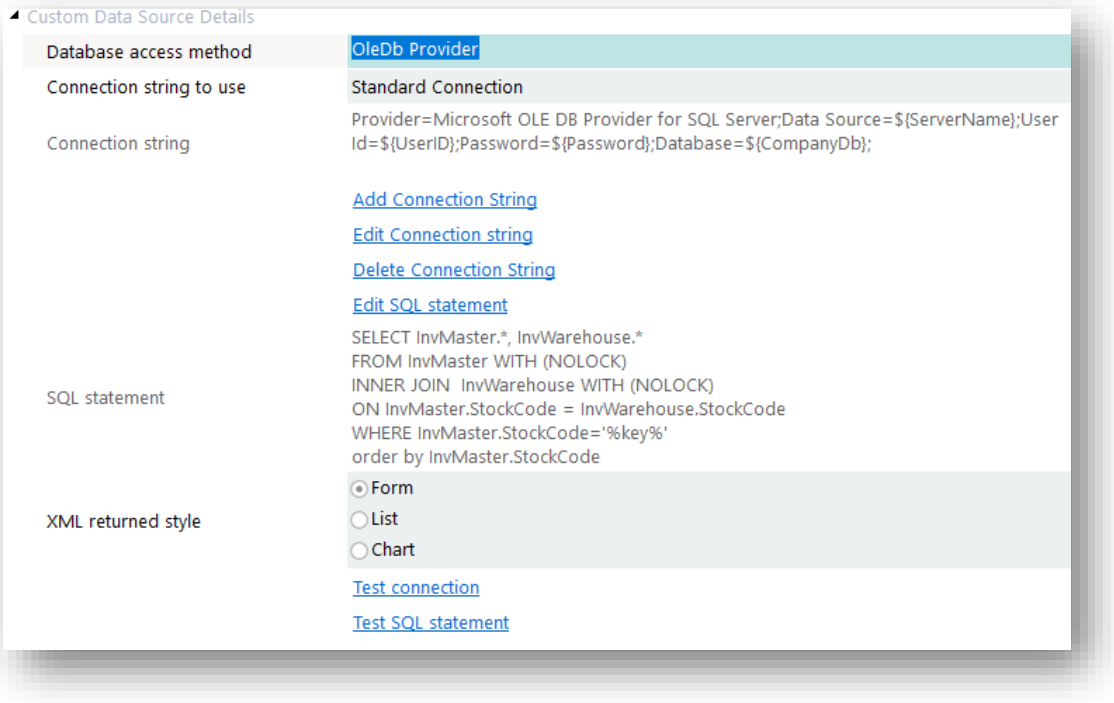
3. Enter a length of 10 and decimals of 0, and then run the application again.
   This time the Credit Limit is shown without decimals and with thousand separators:



You can change any form field in a similar manner, as long as the field is derived from a business object. If you know that a field can contain only a single character value of either Y or N, then you can use the field type of **Check box**.

Custom Data Sources

You can create Custom Data Sources using direct SQL statements and Connection Strings:



You must select the database access method (either OleDb or ODBC) and then define a connection string and SQL statement.

You can execute SQL statements for any type of database, not just a SYSPRO database.

XML in a pre-determined format will be returned no matter which SQL statement is issued. However, the format differs between a FORM style, a LIST style and a Chart Style.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

XML RETURNED STYLE FOR A FORM

The XML is returned in a format suitable for populating a form:

```
<Form><Fields>
  <Field Caption="StockCode" Description="Stock code" Value="A100"
Type="Alpha" Length="30" />
  <Field Caption="Description" Description="Description" Value="15 Speed
Mountain Bike Boys" Type="Alpha" Length="50" />
  <Fields></Form>
```

XML RETURNED STYLE FOR A LIST VIEW

The XML is returned in a format suitable for populating a list view with iterating rows:

```
<Query>
  <Row>
    <StockCode>A100</StockCode>
    <Description>15 Speed Mountain Bike Boys</Description>
    <Warehouse>E</Warehouse>
    <QtyOnHand>950.000000</QtyOnHand>
    <DateLastSale>20150412</DateLastSale>
  </Row>
  <Row>
    <StockCode>A100</StockCode>
    <Description>15 Speed Mountain Bike Boys</Description>
    <Warehouse>FG</Warehouse>
    <QtyOnHand>6.000000</QtyOnHand>
    <DateLastSale>
    </DateLastSale>
  </Row>
  <Row>
</Query>
```

XML RETURNED STYLE FOR A CHART

The XML is returned in a format suitable for populating a Chart with multiple chart points:

```
<Points>
<Point Label='E' Value ='950.000000'/>
<Point Label='FG' Value ='6.000000'/>
<Point Label='N' Value ='909.000000'/>
<Point Label='S' Value ='430.000000'/>
</Points>
```

CONNECTION STRING

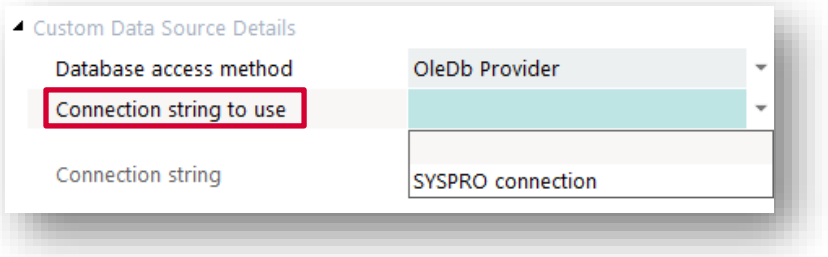
The connection string serves to:

- Identify the server instance and database to connect to
- Determine what driver, login, etc. to use to connect to the SQL Server instance.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

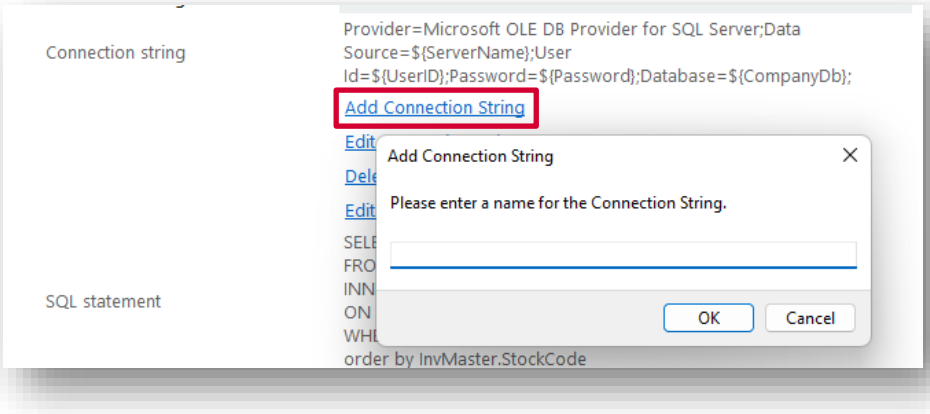
In an application, you can have multiple connection strings to target different databases, and each connection string is given a 30-character name.

For each Data Source you can select which connection string you wish to use from the **Connection String** drop down list:

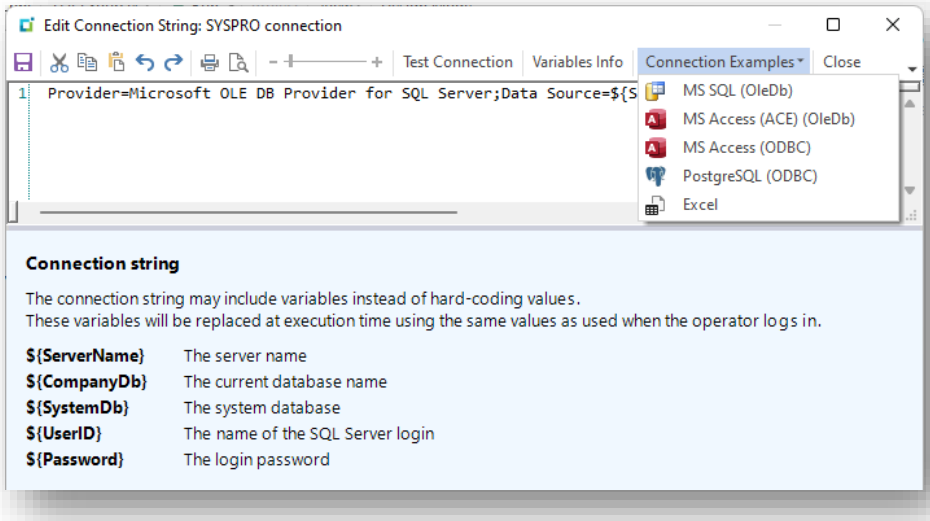


By default, a **SYSPRO connection** string is always supplied. This connection will connect to the same SYSPRO SQL Server instance as used for the operator that's logged into SYSPRO.

Click **Add Connection String** to add a new connection string. You will be prompted to enter a name. The name must be unique within the application.



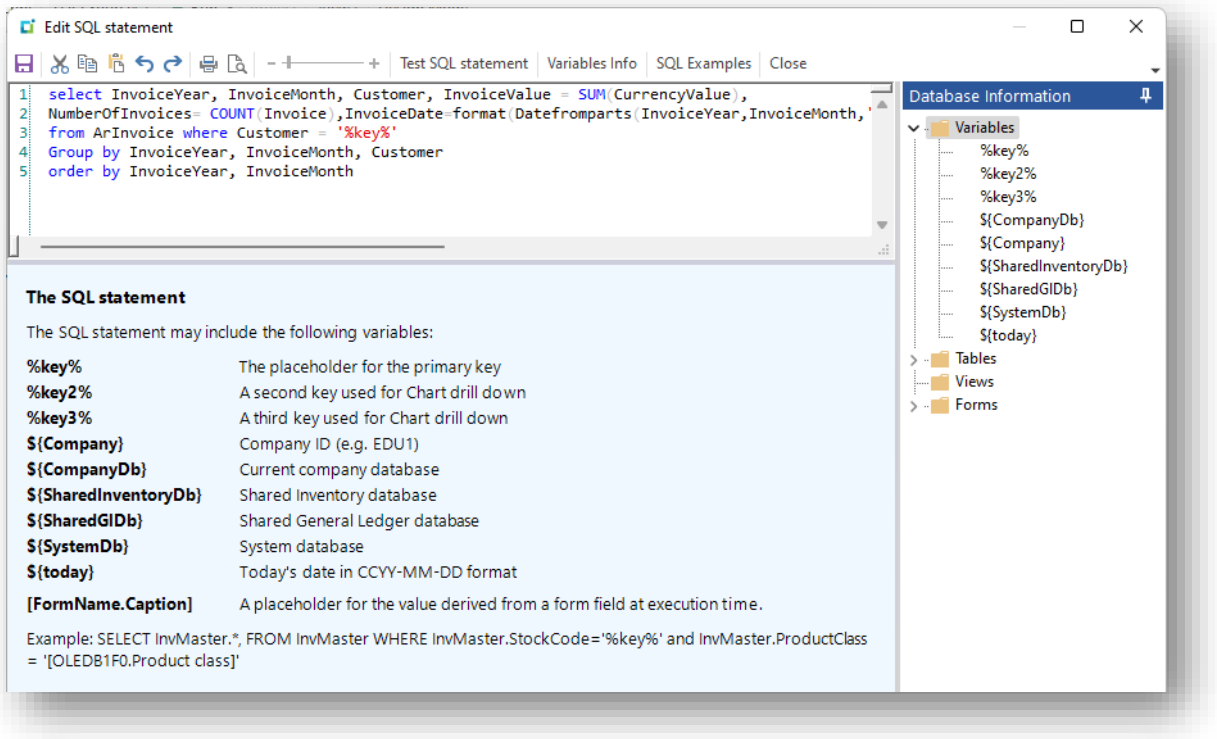
You can now edit the connection string. There are some sample connection strings that you can insert and then adjust as necessary.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Click **Edit SQL statement** to enter a SQL statement.

The tree view in the Database Information pane provides all the variables, Database tables, Database views and Form fields that are available to be used in the SQL statement based on the Connection String. Double-click on any item to insert it into the SQL statement.



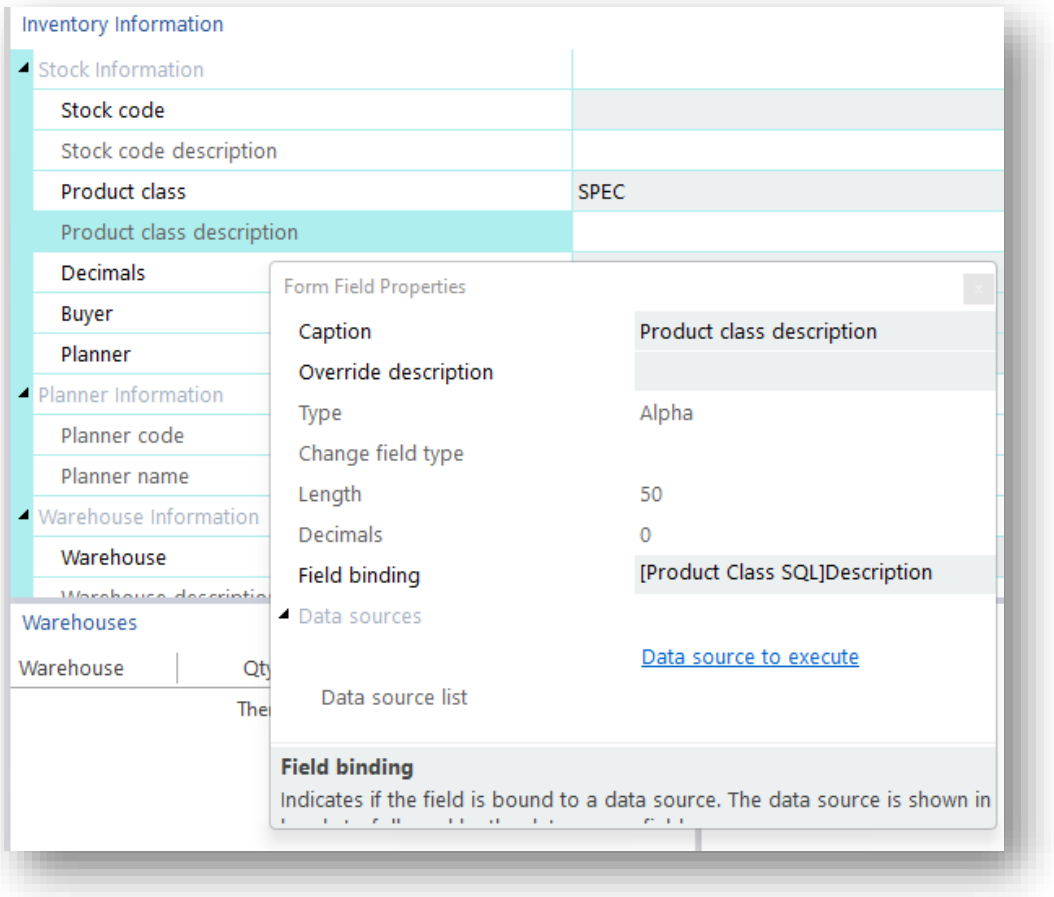
Considerations:

- You can use the variable `%key%` which will be replaced at execution time by the KEY that was used to execute the data source.
- There are other variables that may be used in the SQL statement rather than hard coding for databases or company names.
- There are also two extra Keys (`%key2%` and `%key3%`) that may be used when the data source is to be executed from a Chart drill down.
- You can also use a variable to denote the value of a form field that will be inserted into the SQL statement at execution time.
  The syntax for this is **[*formname.field caption*].**
- You can view the available fields for all forms in the application by expanding the **Forms** item in the tree view. Selecting a form field will inject the correct variable value into the SQL statement.
  For example, this SQL statement:

```
SELECT SalProductClassDes.* FROM SalProductClassDes where
SalProductClassDes.ProductClass = '[OLEDB1F0.Product class]'
```

  will be executed after the current value of the form field **Product class** has been extracted from the form name **OLEDB1F0** in the application:

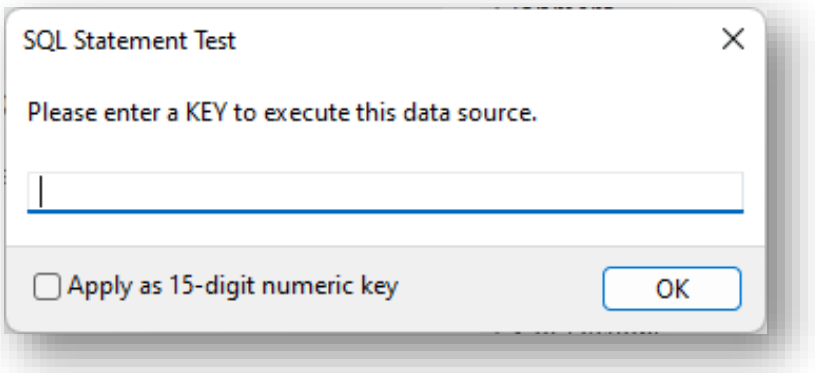
© SYSPRO. All Rights Reserved. All trademarks are recognized.



When you have constructed the SQL statement you can test it by clicking on the **Test SQL Statement**.

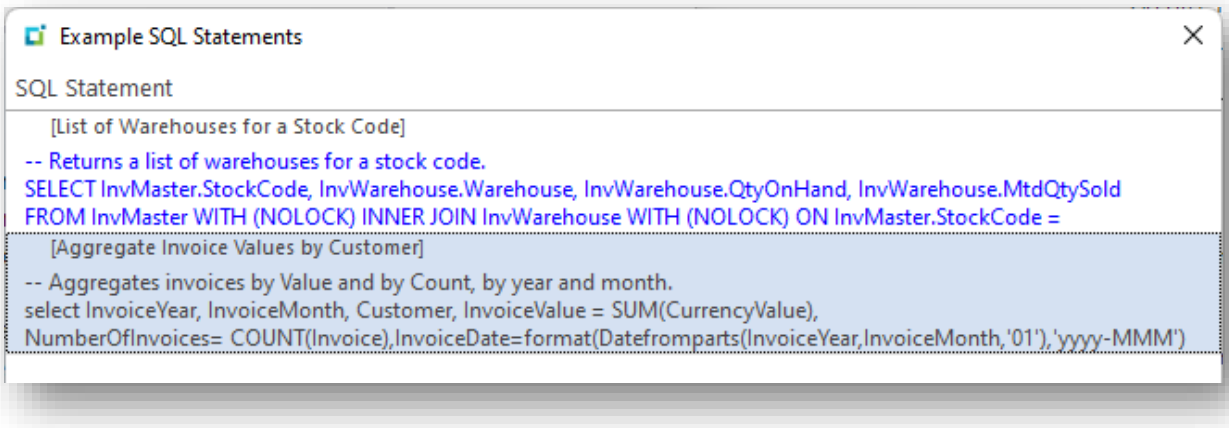
If the statement contains any `%key%` placeholders, then you will be prompted to enter a KEY so that the statement can execute correctly.

If you know that the KEY is a numeric type (for example, for a CUSTOMER numeric key) then you can check the **Apply as 15-digit numeric key** option to save having to enter the full 15-digit number:



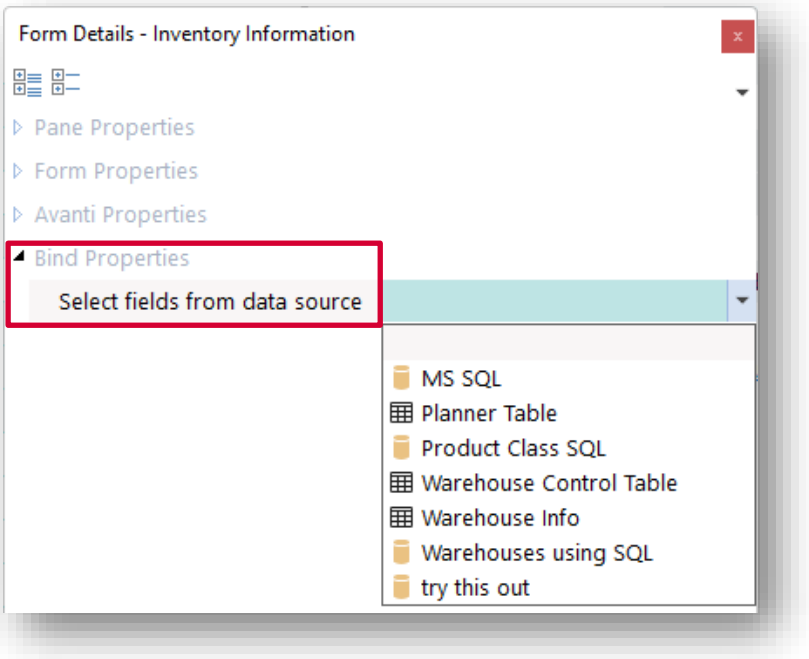
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Numerous examples of SQL statements can be viewed and selected for use. Click on **SQL Examples** and then double-click on any statement for it to be inserted into the SQL editor.



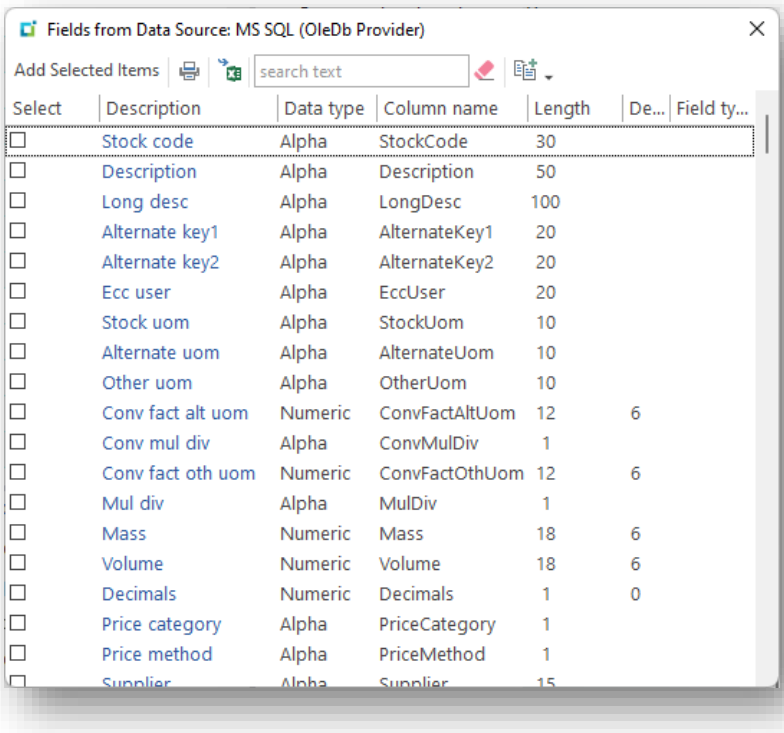
Adding custom data source fields to a Form

Select the data source you want to use from the drop down list in the **Bind Properties**:



The available fields that are derived from the SQL statement will be shown:

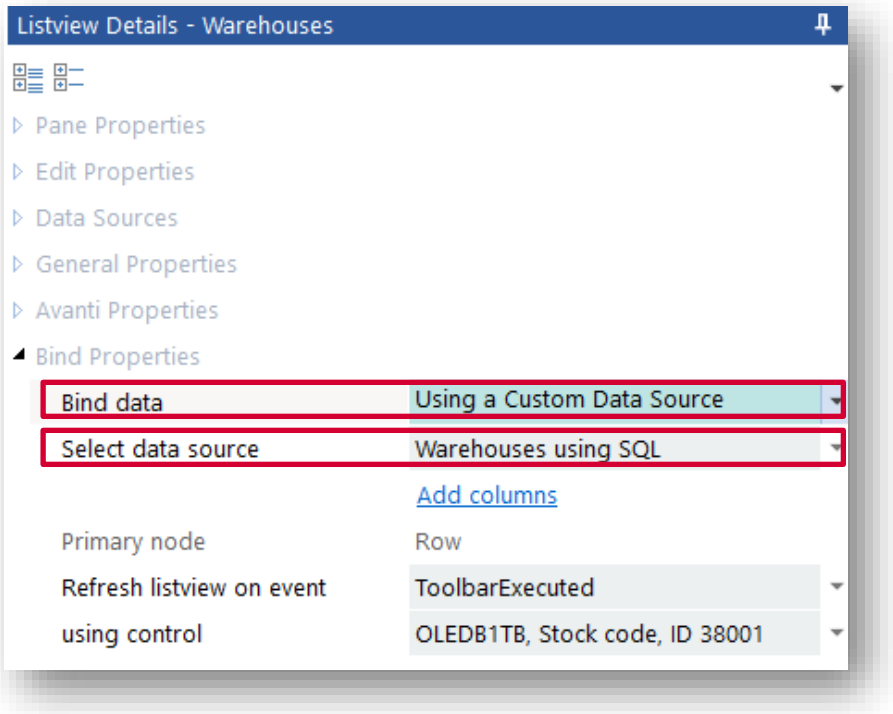
© SYSPRO. All Rights Reserved. All trademarks are recognized.



Select the required fields and click on **Add Selected Items**.

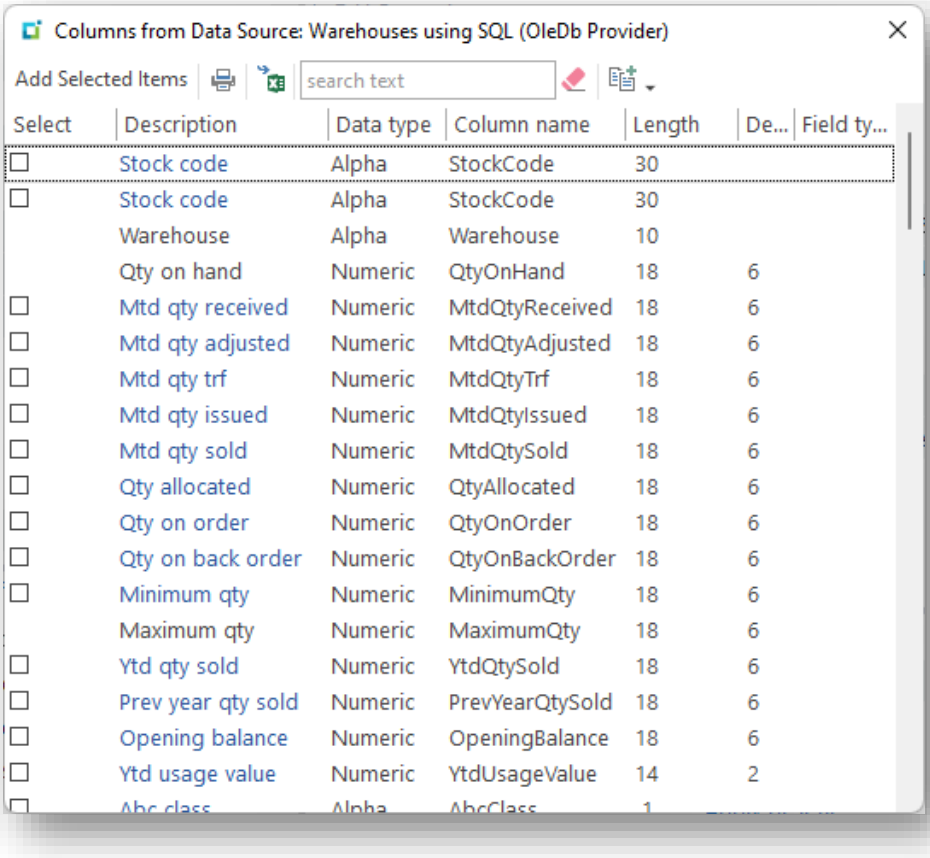
ADDING CUSTOM DATA SOURCE COLUMNS TO A LIST VIEW

Against the list view properties, select the option **Using a Custom Data Source** from the **Bind data** option. Then select the data source from the drop down list.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

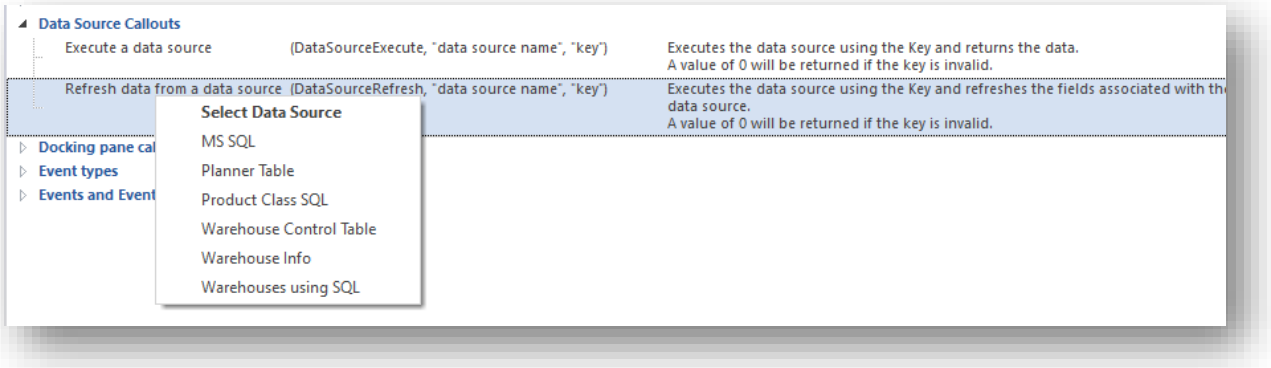
Next, click on **Add columns**. This will show all the columns that are available to be shown in the list view based on the SQL statement:



By default, the PRIMARY NODE will be defined as **Row** and cannot be changed – that's because the data source will always return XML data with **<Row>** elements.

When the custom data source is executed, the list view is automatically populated according to the columns defined for the list view.

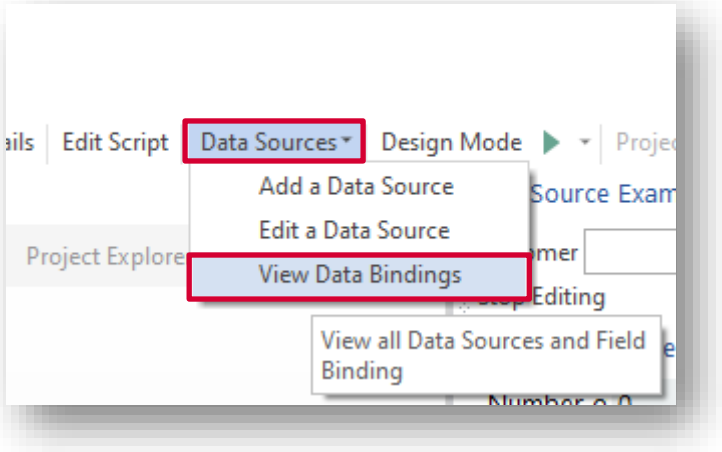
The data source can be executed automatically using the **Refresh listview on event** option, or can be executed directly from the VBScript Data Source callouts:



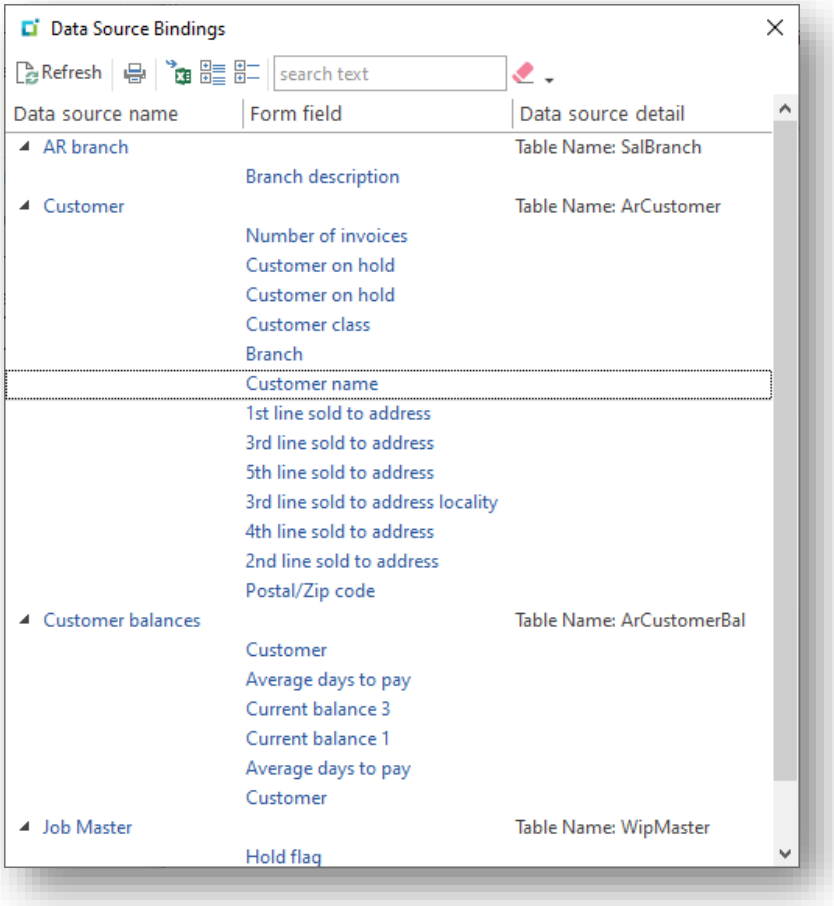
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Viewing Data Bindings

On the main App Designer toolbar, you can view data bindings:



This will display all the form fields that are bound to data sources.



Click on the Data source name hyperlink to edit the data source.

Click on the hyperlinked form field caption and this will navigate to the form field.

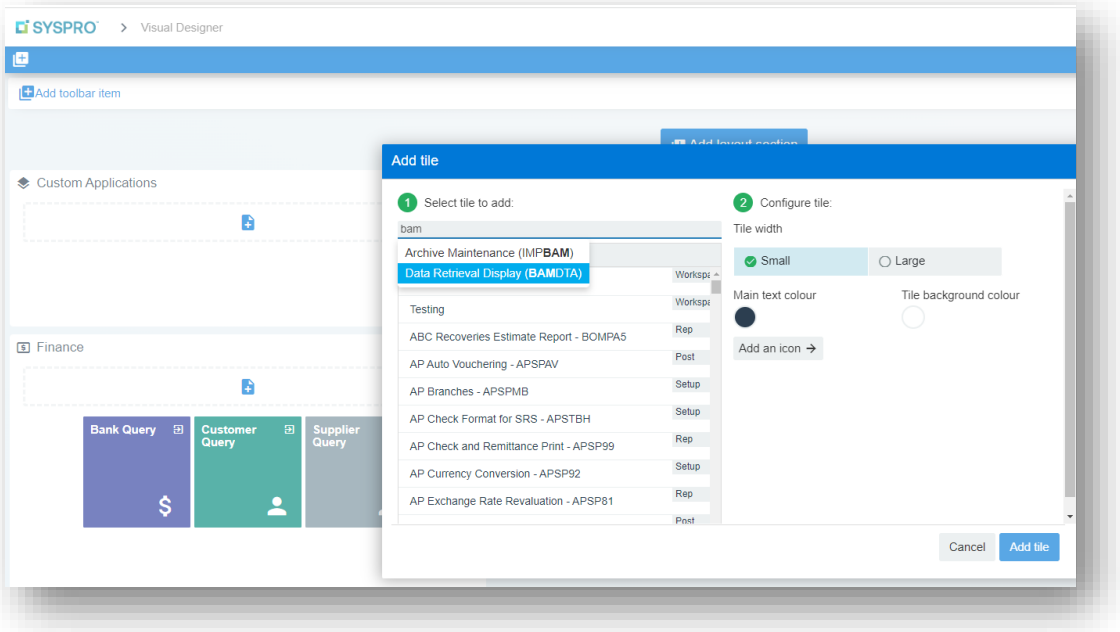
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Adding Custom Applications to the Desktop or Web UI Menu

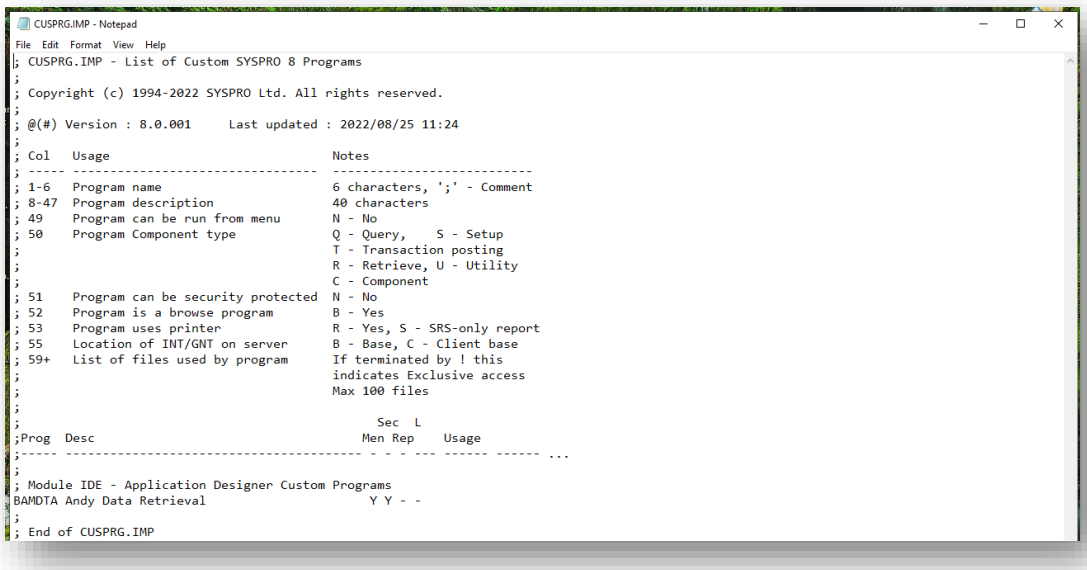
Custom applications can be run as you would any typical SYSPRO program, but you may wish your application to appear on the **Program files** menu (in the Desktop UI) or the **Hamburger** menu (in the Web UI Avanti) just like a SYSPRO program.

The following steps explain how to achieve this:

1. Add the application name to the text file `CUSPRG.IMP`.
   This entry makes your application appear when adding tiles in Avanti:

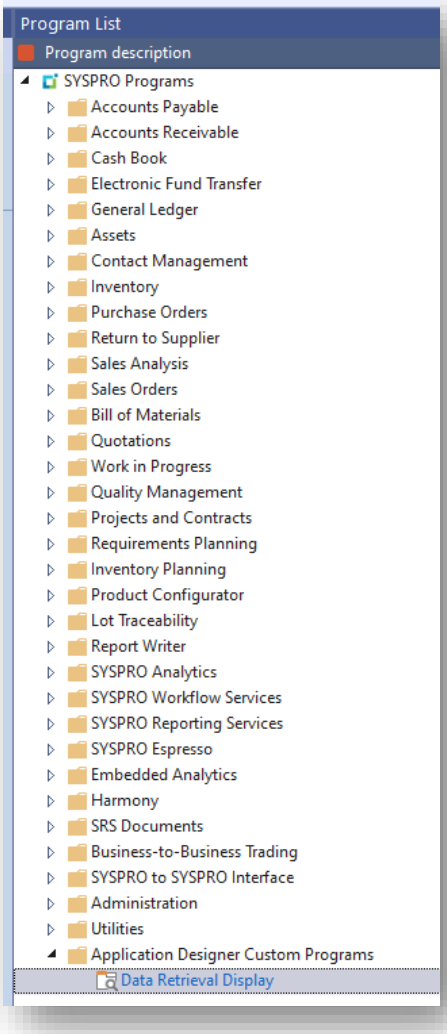


   The format for `CUSPRG.IMP` is identical in structure to `IMPPRG.IMP` (found in the `…\base\store` folder):

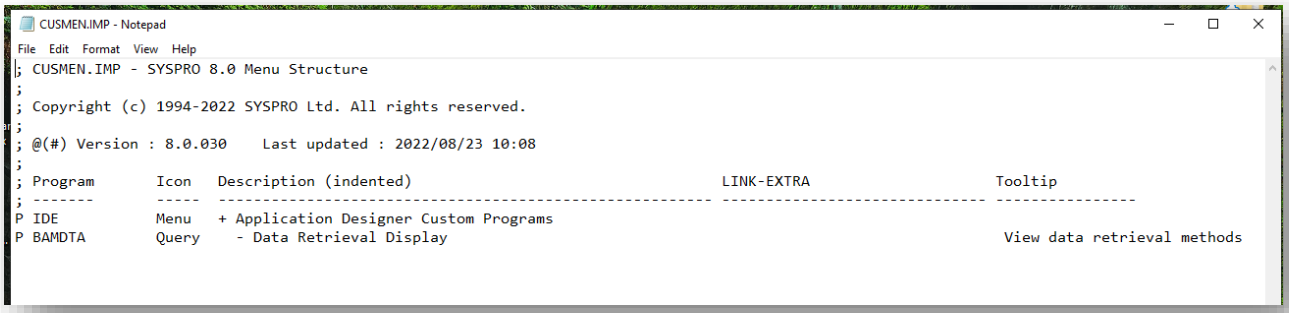


© SYSPRO. All Rights Reserved. All trademarks are recognized.

2. Add the application name to the text file `CUSMEN.IMP`.
   This entry makes your application appear in the Program List:



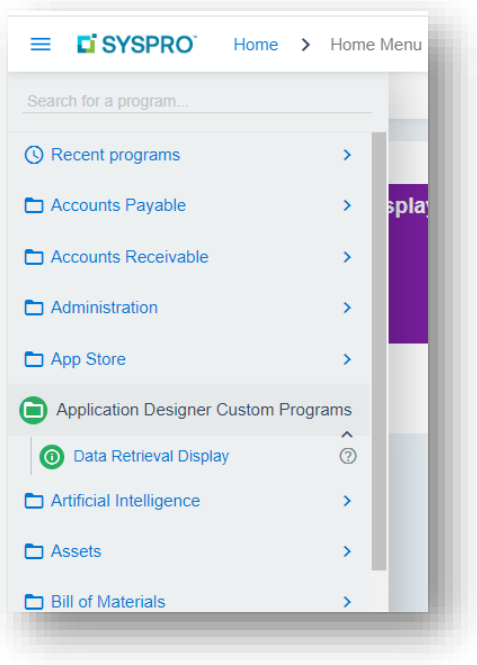
   The format for `CUSMEN.IMP` is identical in structure to `IMPMEN.IMP` (found in the `...\base\store` folder):



Note: These files can be found in the `Plugin\CustomStore` folder.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Tip: In Avanti, custom programs and modules on the Hamburger menu are indicated with green coloured icons:



Getting Help

This guide, the Help panel built into the Designer, plus the **Application Designer** movies will get you up to speed with designing and deploying applications.

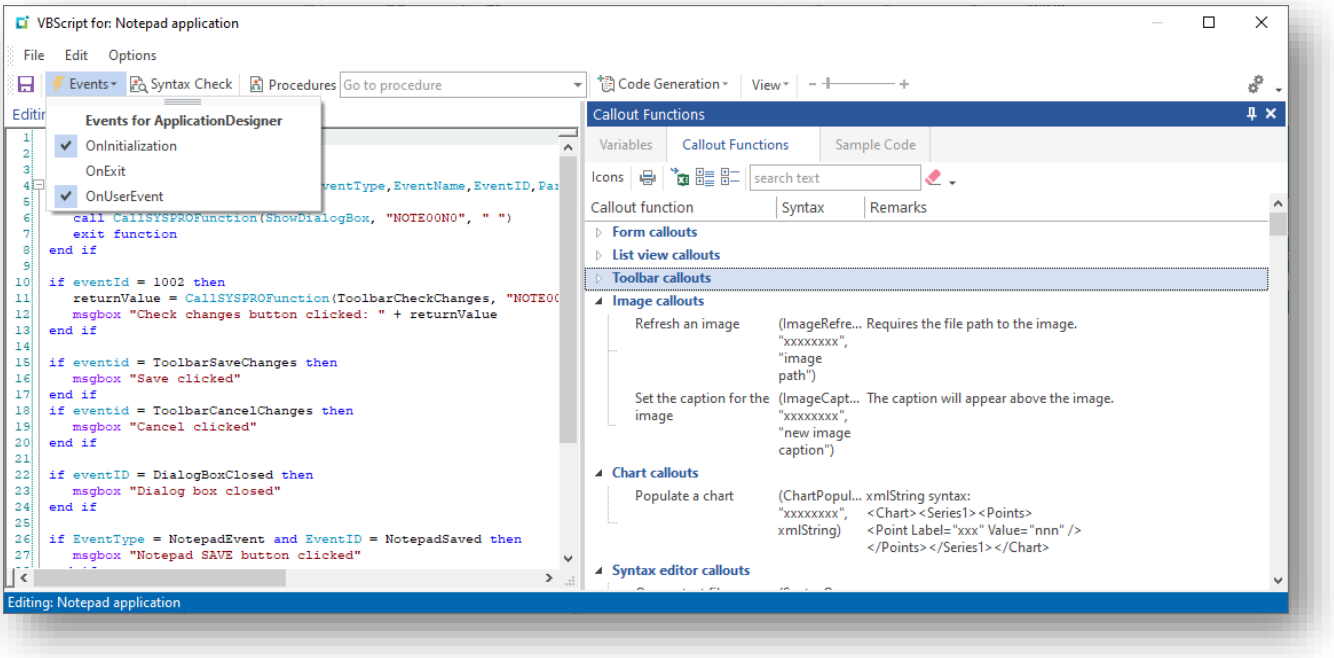
You may also consider visiting our Forums page at https://forums.syspro.com/ .

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Programming the Application

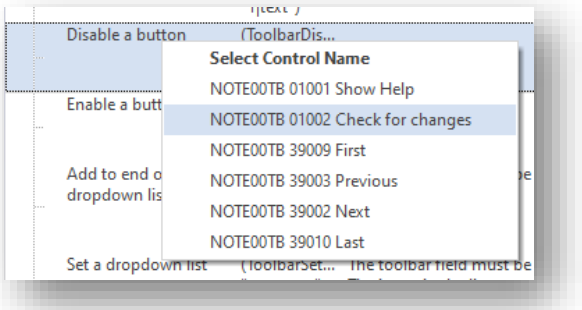
The application's behavior and logic can be controlled by script. The script contents are embedded in the application when the application is saved.

To edit the script, click on the **Edit Script** button in the main toolbar. The script editor window will look something like this:



The editor consists of a main text control to enter the script and a variety of other panes that help create script content. The main panes are covered here:

1. **Callout functions**
   This pane contains all the callouts that can be invoked by a script. These callouts affect the control and logic of the entire application, from disabling toolbar fields to populating a data grid.
   To apply any callout, simply double-click on it and a context menu will popup showing all the controls that are appropriate for that callout:



   Click on the required control name and the callout statement will be injected into the script editor at the current text position.

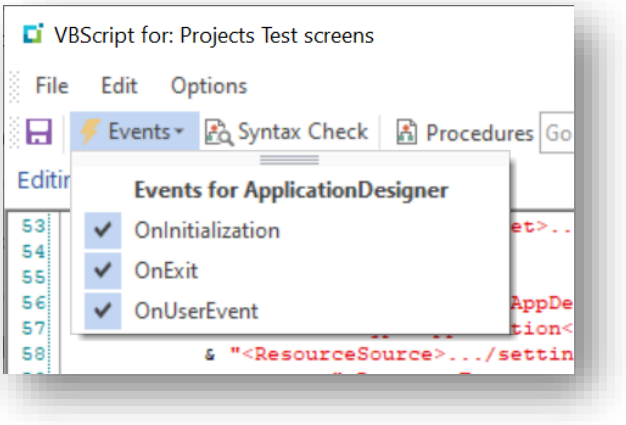
© SYSPRO. All Rights Reserved. All trademarks are recognized.

2. **Variables**
   These contain various read-only (and occasionally read-write) variables that you can use in your code, anything from the current company code to various folder paths.

3. **Sample code**
   This pane contains example code that can be inserted into the script.

Application Designer VBScript Events

An application can raise one of three events each time the UI is instructed to do so:



`OnInitialization` and `OnExit` events do not take any parameters and are raised when the application is initialized and exits respectively.

Scripting the OnExit Event

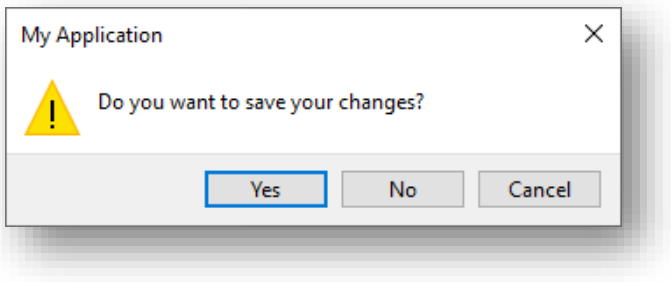
The `OnExit` event can be used to prevent a user from inadvertently exiting out the application by setting the `AppDesigner_OnExit` function to be **false**.

Here's an example of how to use this:

```
AppDesigner_OnExit=False
returnValue = CallSYSPROFunction(TaskDialog, " ", "<Msg><Title>My
Application</Title><Text>Do you want to save your changes?</Text><Footer Icon=''
Text='' /><Options /><Buttons Default='002'></Buttons><Icon>warn</Icon><Standard
Cancel='true' Yes='true' No='true' /></Msg>")
dim TaskDialogArray
TaskDialogArray = split(returnValue,vbTab)
if TaskDialogArray(0) = 2 then
   AppDesigner_OnExit=true
end if
End Function
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

When this application runs and the user tries to close the application, the following message will appear:



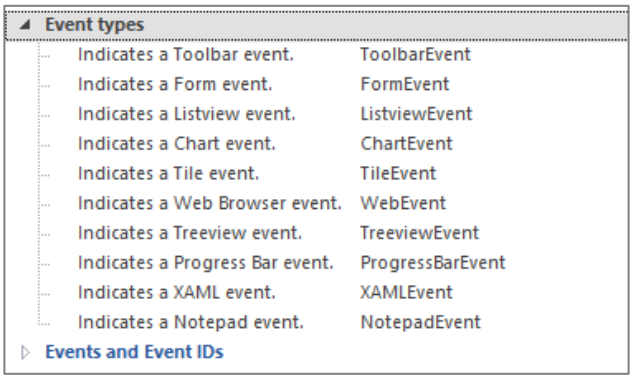
If the user selects **Yes**, then the application will exit, otherwise it won't.

SCRIPTING THE **ONUSEREVENT** EVENT

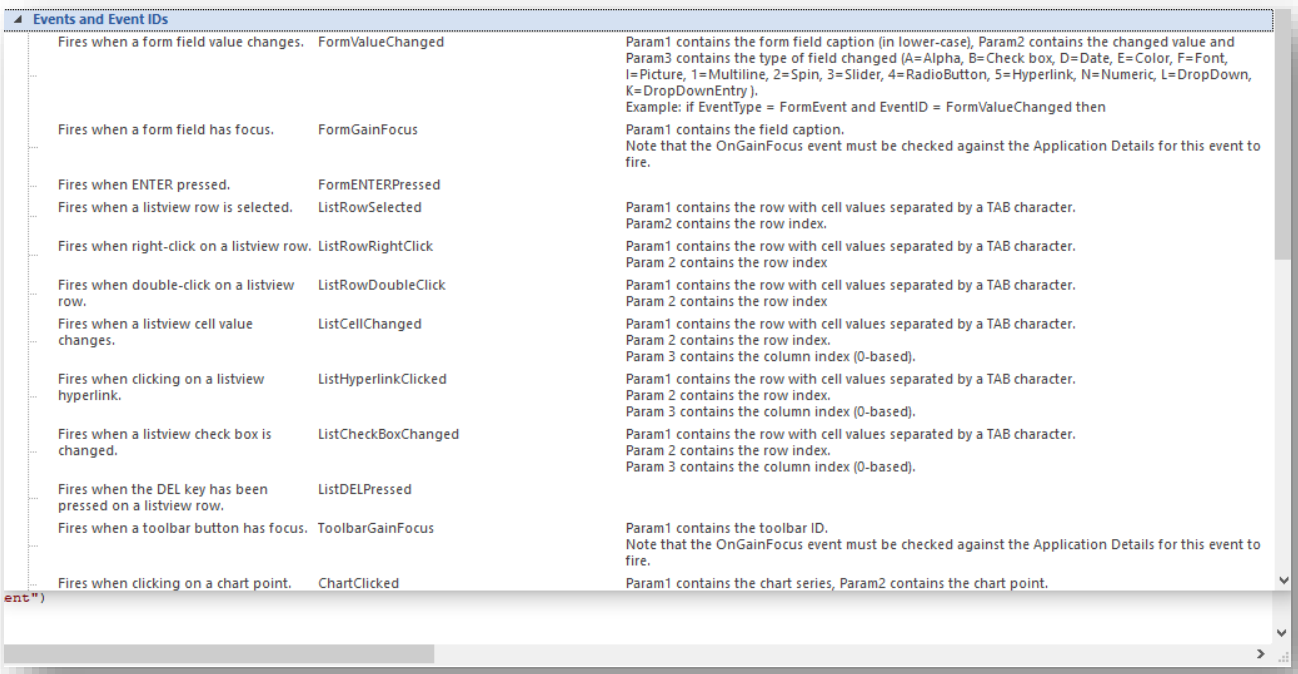
The most used event is the **OnUserEvent**, which is raised any time a user interacts with the UI. This event has the following format:

```
Function AppDesigner_OnUserEvent(EventType, EventName, EventID, Param1, Param2, Param3)
```

These parameters are explained as follows:

Parameter	Explanation
EventType	This indicates the type of control that invoked the event. These event types can be seen in the Callout pane of the VBScript editor: 
EventName	This is usually the name of the pane that the control is on.
EventID	This is the unique id of either the toolbar button executed or otherwise the specific type of event. You can view all the events that can fire in the Events and Event IDs section in the Callout pane – see sample image below.
Param1,2,3	These hold the data relevant to the type of event.

© SYSPRO. All Rights Reserved. All trademarks are recognized.



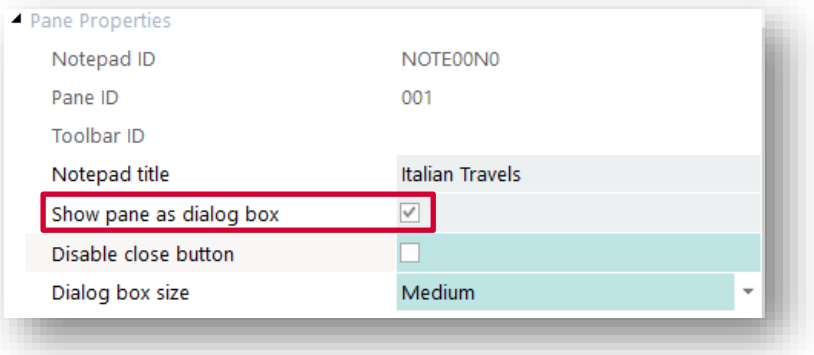
To add an event into the VBScript, select **Events** from the toolbar menu, followed by the event that you need. It is logical (and sensible) to put the events in sequence in your script (e.g. `OnInitialization`, `OnUserEvent`, `OnExit`).

Note: VBScript events only execute when the application is running. They do not fire when you are designing an application.

How to use a Dialog Box

Dialog boxes in an application can be a useful way to present information in a popup window that the user must then close to continue processing.

In the Application Designer, any control and its associated toolbar can be presented in a dialog box. To make a control appear as a dialog box, enable the **Show pane as dialog box** option:



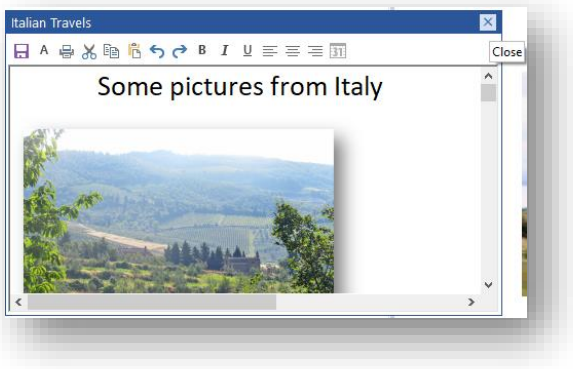
© SYSPRO. All Rights Reserved. All trademarks are recognized.

This option ensures that this pane will be automatically hidden (and disabled) as the application loads, regardless of how the pane appears while designing the application.

The only way to then show the dialog box is via code, using the script callout **`ShowDialogBox`**. When this callout is invoked, the pane is shown as a modal window and interaction with the main application is prevented until the dialog box is closed.

There are two ways a dialog box can be closed:

1. By the user clicking on the **X** in the top right-hand corner of the window:



2. By using the script callout **`CloseDialogBox`**.
   You may wish to prevent the user from exiting out of the application by clicking the X, so you can select the option **Disable close button**. This means that the only way a dialog box can be closed is via code.
   Typically, you would have a CLOSE or SAVE button in the control's toolbar and have code behind these buttons that then invokes the **`CloseDialogBox`** function.

Note: When a dialog box is closed the event **`DialogBoxClosed`** fires. You might use this event to set focus to a form or toolbar.

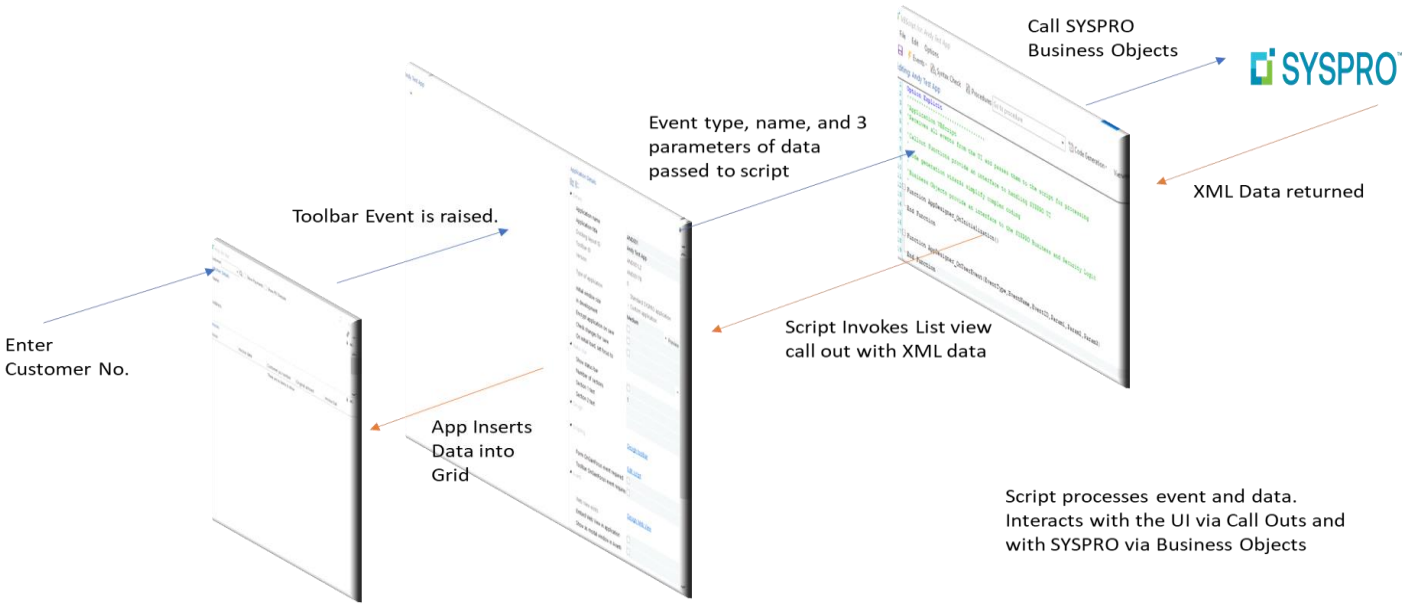
You can also define the initial size of the dialog box with the option **Dialog box size**. This is just the initial size to be presented to the user; thereafter the system will remember the position and size of the dialog box.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

EVENT CYCLE PROCESSING

This diagram shows how events are typically processed in an application.

1. The user enters a Customer Number in a toolbar.
2. A toolbar event is raised which is intercepted by the application's script; the script calls a business object which in turn returns XML data.
3. The script processes the XML data and issues a callout to populate a list view with the XML data.
4. The list view is populated with data, the script exits, and control is passed back to the application.



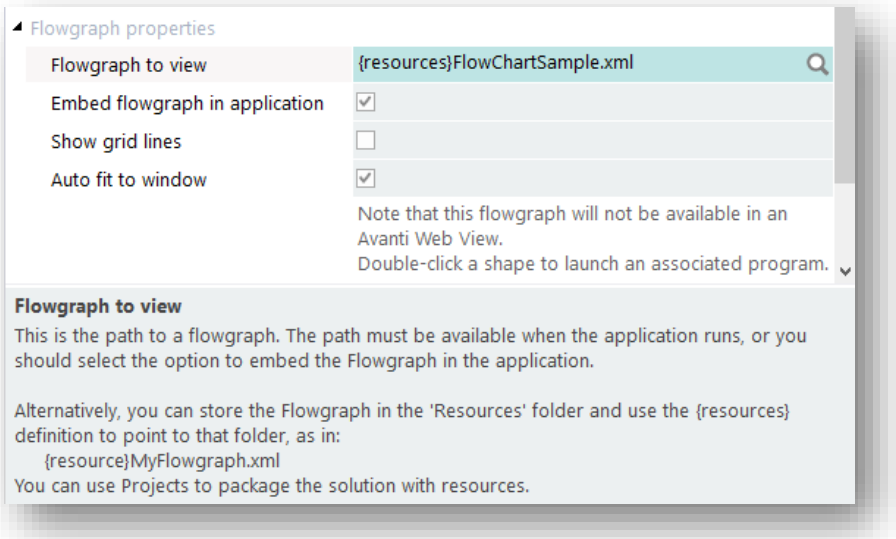
© SYSPRO. All Rights Reserved. All trademarks are recognized.

SOME BASICS

Important information regarding application file names and folder locations:

- Applications have a 6-character application name.
- Applications reside as text files in the SYSPRO …/base/settings/AppDesigner folder.
- Project files reside in the …/base/settings/AppDesigner/Projects folder.
- There is a …/base/settings/AppDesigner/Resources folder which can be used to hold any special resources that the application may need.
- In a client/server installation, the Applications in the AppDesigner folder and the Resources folder are synchronized with the main server.

  Note: The Projects folder is not synchronized: this is a personal development tool for a developer and as such, resides on the local machine of each developer.
- Hard coded folder names should be avoided, and relative folder names should always be used. In this way the application does not become bound to a specific SYSPRO installation.
- The VBScript editor exposes the following useful System Variables that help construct relative folder names:
  - BaseSettingsFolder
  - Company
  - MultiMediaPath
  - VBSModulesPath
- The Application Designer also exposes a special keyword syntax for pointing at the Resources folder using the keyword **{resources}** in the path for various controls:
  - Flowgraph
  - Notepad
  - Image viewer
  - Document viewer
  - Syntax Editor
  - XAML control



© SYSPRO. All Rights Reserved. All trademarks are recognized.

- The **{resources}** keyword can also be used in VBScript when setting a control's filename.

```
Function AppDesigner_OnInitialization()
    dim filename
    filename = "{resources}al00.pdf"
    call CallSYSPROFunction(DocumentShow, "NOTE00P0", filename)
End Function
```

Note: The interpretation of **{resources}** is done by the control not by the VBScript.

- The operating system temporary directory can be used to contain temporary files (see later).
- An application named **SAMXML** will generate a file called `SAMXML_AppDesigner.txt`. It is important therefore that a naming convention is established so that if applications are being written by multiple developers, there is no clash in application names. This is particularly important when using the project export and import function. It would be very easy to overwrite an application written by someone else.

  This goes for any component such as .Net User Controls, and VBS modules.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Some VBScript Suggestions

This section tries to provide pointers to what we have found to be good practice.

Because the Application Designer is so easy to use and writing VBScript is not onerous, thought should be given to other developers who may either need to enhance the code or need to debug it should there be a problem.

Functions and Classes

- Write code in re-useable functions and pass parameters to them.
- Do not write monolithic pieces of code. They will be difficult to debug later.

Comment the Code

- Introduce the code with a description of who wrote it, when and what it is supposed to do.
  For example:

```
Option Explicit
''''''
```

APPDESIGNER_ONUSEREVENT

This event is the most critical of events, as every control will raise this event when actions are performed on them.

It is useful to be able to see what data is being passed into the event by a control. Therefore, the following code is an example of how to do that:

```
dim msg
msg = ""
msg = msg + "EventType: " + EventType + VBCRLF
msg = msg + "EventName: " + EventName + VBCRLF
msg = msg + "EventID: " + EventID + VBCRLF
msg = msg + "Param1: " + Param1 + VBCRLF
msg = msg + "Param2: " + Param2 + VBCRLF
msg = msg + "Param3: " + Param3 + VBCRLF
msgbox msg
```

See Appendix for code snippet.

THE USE OF CONSTANTS

The Application Designer allocates unique EventIDs to Toolbar controls, which are then passed back the VBScript. Using these EventIDs, the VBScript can identify what the control was and make appropriate actions.

*See above for the **OnUserEvent** parameters.*

It is important to structure the code in such a way as to make it easy to read. Therefore, allocating constants with sensible names that refer to the toolbar button EventIDs helps readability and maintenance.

For example:

```
23  'toolbar event ids to help make code more readable
24  const TBar_Help    = "01001"
25  const TBar_DataRetrievalMethod    = "01002"
26  const Tbar_CustomerNumber  = "38000"
27  const Tbar_ShowDebugMessages = "01015"
28  const Tbar_WriteXMLtoFile = "01016"
29
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

Catching the **EventType** and then using a select statement to catch which control was pressed works well:

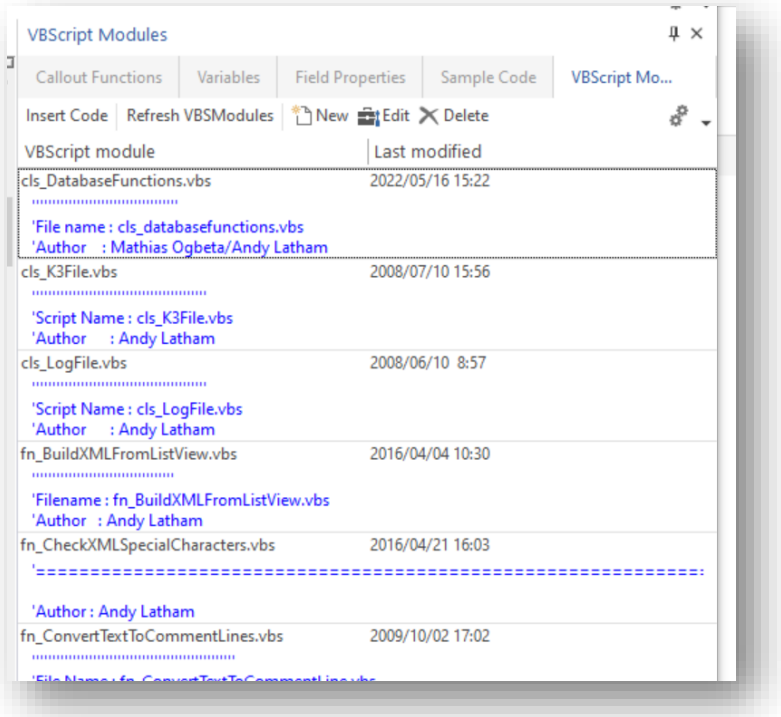
```
'Catch toolbar events
if EventType = ToolbarEvent then
  select case EventID

  '****Customer selected in combobox
    case Tbar_CustomerNumber
      CustomerNumber = param1
      call RetrieveData(CustomerNumber, dataretrievalmethod, sqlserver,sqldb

    '****show Debug Messages checkbox clicked
    case Tbar_showDebugMessages
         chkstatus = param1
         if chkstatus = 1 then
             showDebugMessages = true
         else
             showDebugMessages = false
         end if
```

See Appendix for code snippet.

RE-USEABLE VBSCRIPT FUNCTIONS AND CLASSES

When developing several applications, there will inevitably be script functions and classes that are used across applications.

Try to avoid cutting and pasting common code from one application to another and use the VBS modules functionality within SYSPRO. The **LoadModule** function will inject the code into the VBScript before it is executed.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

If you double-click on a VBS module, then this code is inserted.

At run time, the code in the module will be inserted exactly where the **LoadModule** call is made.

```
LoadModule("cls_LogFile.vbs")
```

Debugging and Logging Progress

The Application Designer does not have a VBScript debugger that will allow stepping through of code to check logic and contents of variables.

The best way to do this is using either a msgbox command to display data, or the Application Designer Task Dialogue box (see Sample App `SAMTSK` for examples).

As well as displaying progress, it may be useful to write out a data string (e.g. XML or XAML) being built so that it can be copied into other programs for testing syntax.

For example:
Generating XAML for a XAML pane, output the constructed XAML and use Visual Studio Blend to see what it looks like.

Please read the documentation on the XAML control for more information.

In both cases, it is often worth putting a flag in the code to be able to turn logging on and off. If it is in place to help develop the App it may be required later to debug it and often by other people so, make it easy for them.

Remember, if writing out files, make sure that the folder name is either relative to where the application is run or use SYSPRO variables to construct it.

For example:

```
filename = SystemVariables.CodeObject.baseSettingsFolder & "SAMDTA_ADODB_XML.XML"
```

Or

```
35 Set fso = CreateObject("Scripting.FileSystemObject")
36 tempdir = fso.getspecialfolder(2)    'gets local temp directory
```

See Appendix for code snippet.

Also check VBS modules for a class called `cls_LogFile.vbs`.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

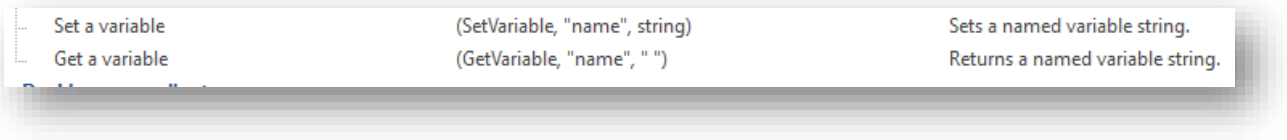
Using Persisted Variables

Remember that the VBScript events are raised by the UI each time a control is interacted with. This means that the VBScript is run each time an event occurs. It is therefore not possible to set a variable, give it a value when one event occurs and then read it when another event occurs. All variables will be initialized each time the script is run.

There are two ways of setting variables that persist for the duration of an application:

1. Using system global variables:
   There are 4 Global variables in the System variables pane that can contain string data and that are accessible during the run of SYSPRO.
   You should use this carefully as such variables can be used by other applications and may overwrite your variables.

2. The preferred method is to use the VBScript callout functions **`SetVariable`** and **`GetVariable`**.
   During the execution of a script, you may want to save a string as a variable and have that variable available during another script execution during the same run of the application.
   This is achieved using the callout **`SetVariable`**.
   The callout allows you to name a variable (up to 8 characters long) and to provide a string of indeterminate length.

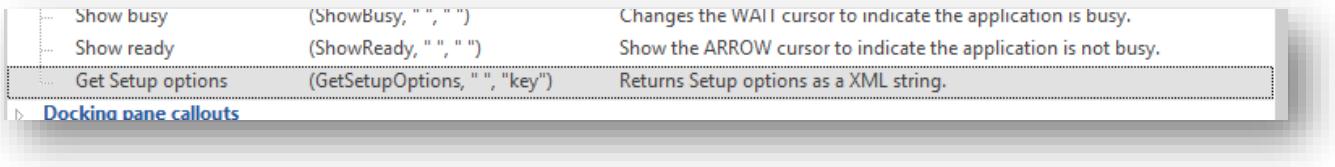


   You can have as many variables as you like. When the application has exited, the variables used for that application during the run of the application are removed. You can at any time retrieve the variable string using the variable name with the callout **`GetVariable`**, in which you just supply the variable name, and the returned value is the variable string.

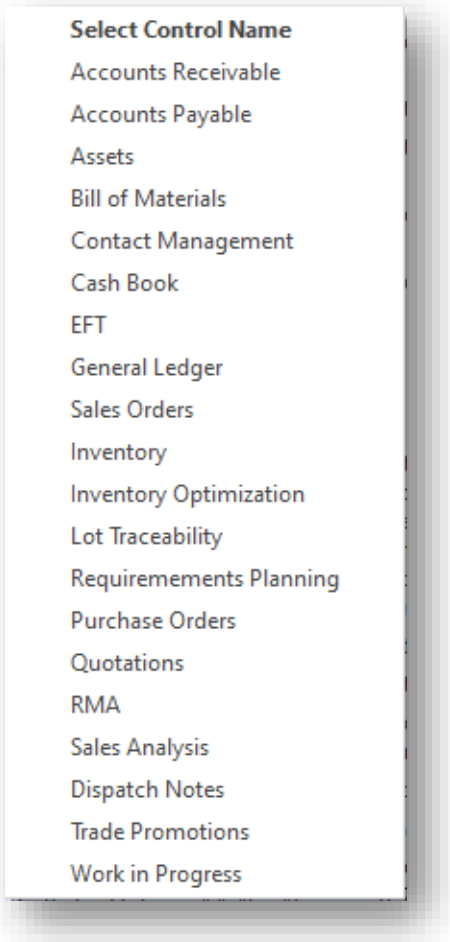
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Retrieving Setup Options

You may need to get access to setup options to help determine the behaviour of the application. While it is possible to call any of the Setup business objects from a script, this process is simplified with the callout **`GetSetupOptions`** in the script editor:



Double-clicking on this item will display a list of modules:



Select the required module (e.g. Inventory) and a statement will be inserted into the script editor like this:

```
returnValue = CallSYSPROFunction(GetSetupOptions, " ", "INV")
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

On execution of this statement, **`returnValue`** will contain an XML string of options:

Using an XML parser, you can now retrieve the option(s) as required.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Understanding the Controls

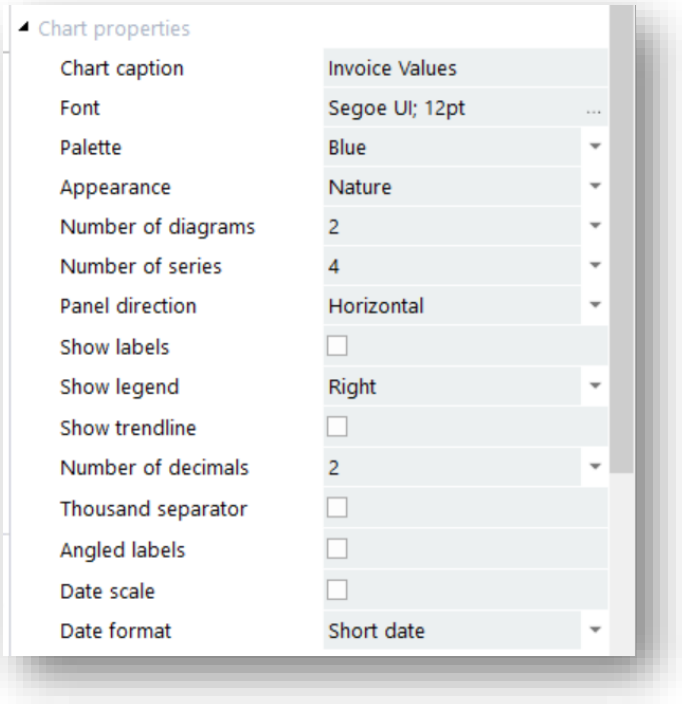
In this section we explore how to make best use of the various controls that make up an application.

CHART CONTROL

The chart control is a powerful, easy to use control with numerous ways to display data.

The look and feel options are defined when setting up the Chart pane in the Designer, leaving the developer to focus on constructing XML data to send to the chart.

MAIN CHART OPTIONS



Using the control is straight forward. The Callout functions below explain the functionality available.

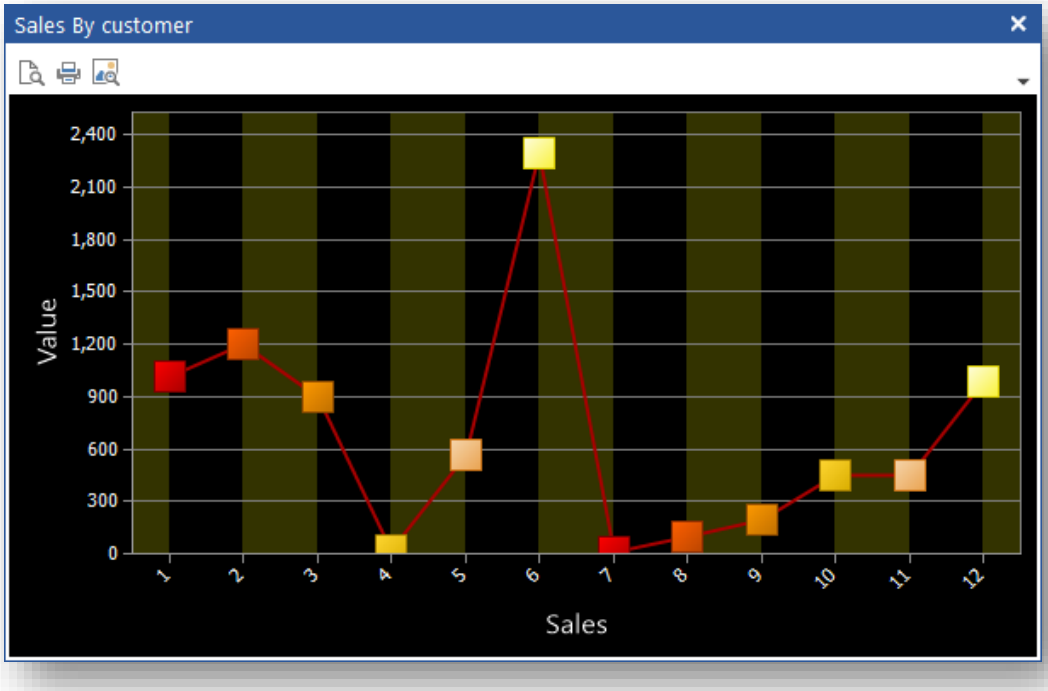
FEATURES OF THE CONTROL

There are several options to control the appearance of each diagram in a chart.

You may have up to 4 diagrams in one chart, and up to 4 series.

Each series can be in separate diagrams or grouped within a selected diagram.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

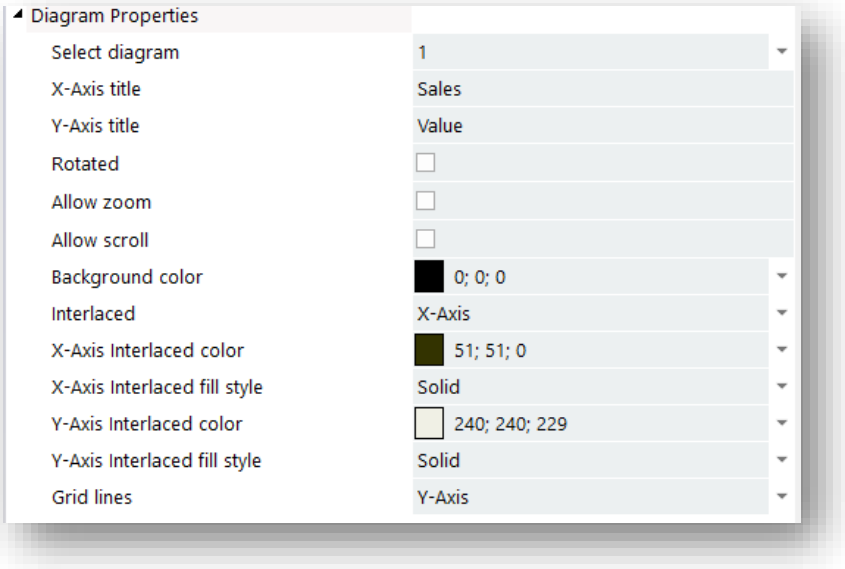


© SYSPRO. All Rights Reserved. All trademarks are recognized.

The properties of a diagram include:

- Interlaced colors
- Grid Lines
- Background colors
- Transparency option for Area and Bubble styles

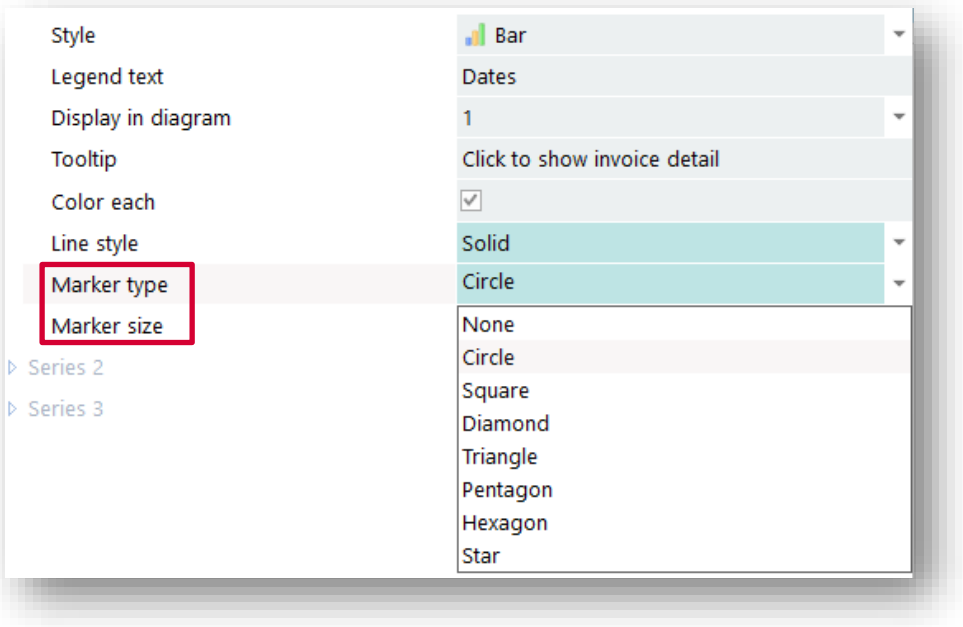
You can also rotate the diagram and allow the user to zoom into a chart area and then scroll the chart.



Additionally, you can show a **Secondary Y-Axis** for the chart. A secondary axis is useful when you need to display a series of data of values that greatly differ from each other, or you might use a secondary axis when displaying two series of data with different data type (i.e. price and quantity). In both cases you can plot each of the series on a secondary axis so that they can both be seen in the same view:

© SYSPRO. All Rights Reserved. All trademarks are recognized.

The properties for each series include Marker Type and Size:

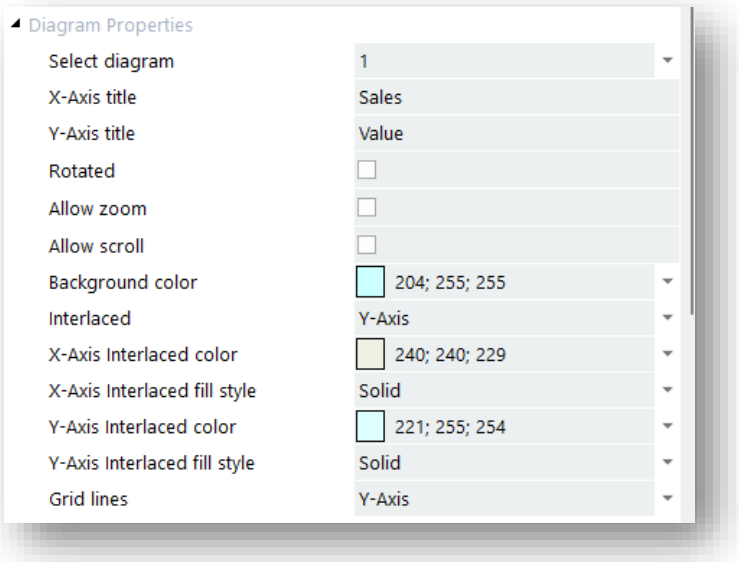


Each series can be displayed as a different chart style.

There is no limit to the amount of data (data points) that can be sent to a chart and a mini toolbar allows for print preview, printing and displaying in an image viewer.

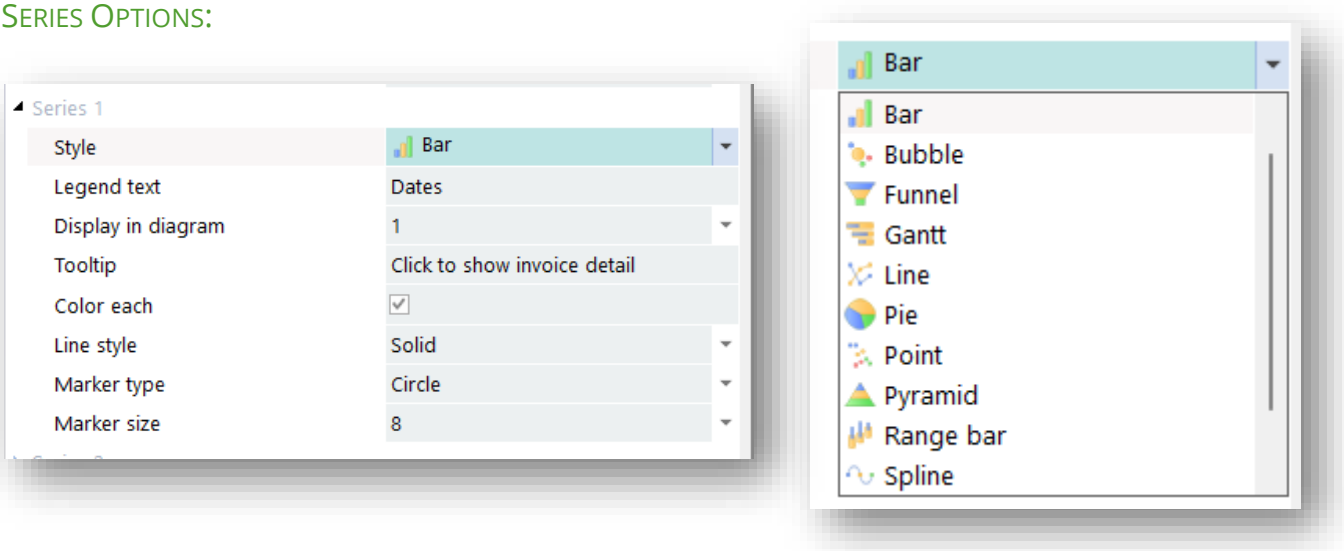
For example:

DIAGRAM OPTIONS:

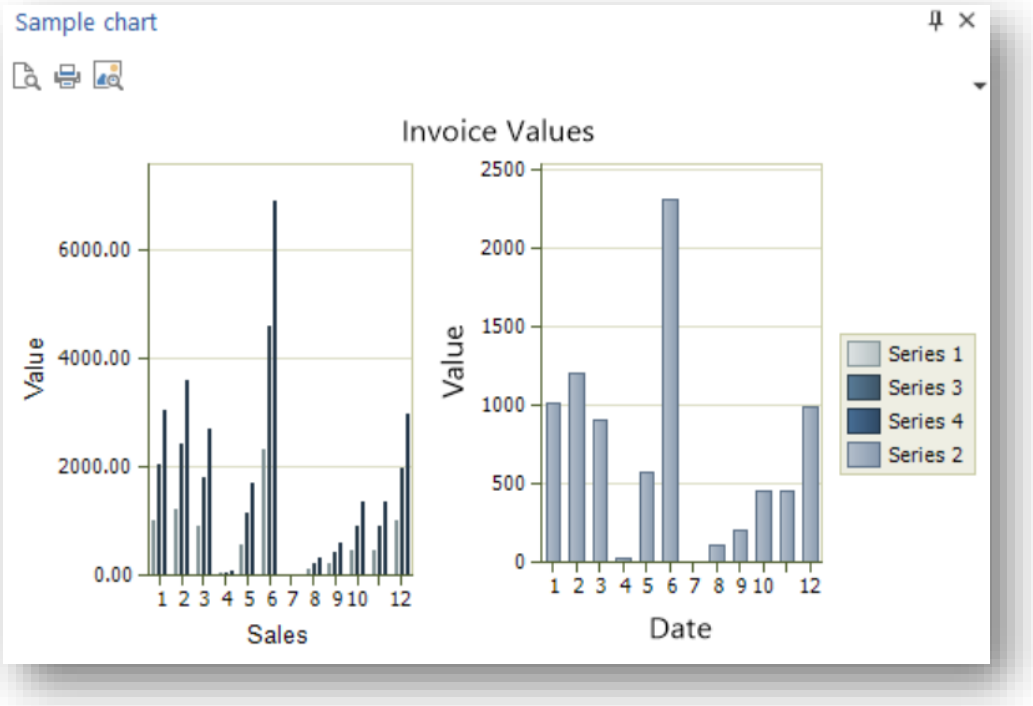


© SYSPRO. All Rights Reserved. All trademarks are recognized.

SERIES OPTIONS:



OUTPUT:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

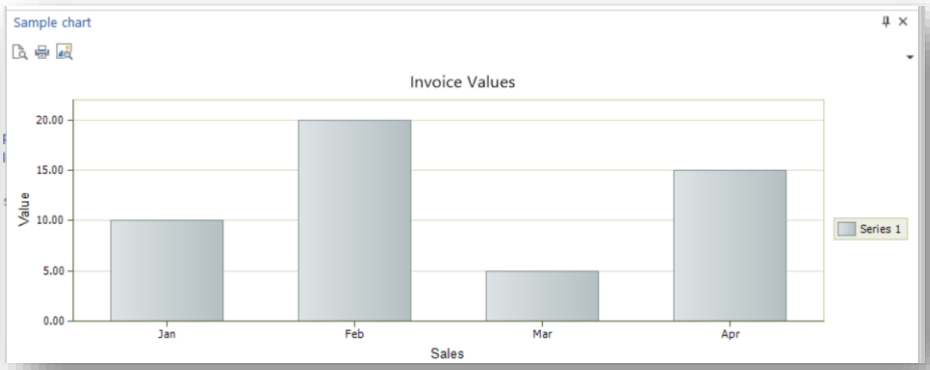
CALLOUT FUNCTIONS

The Application Designer has the following callout functions for the Chart control:

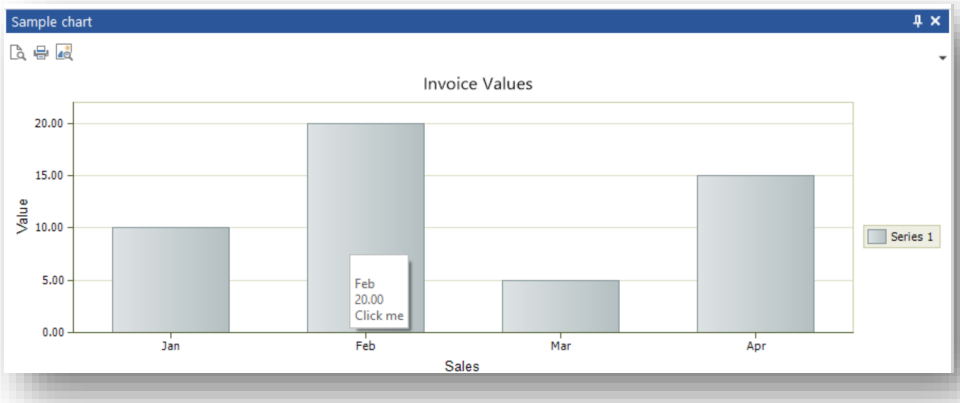
Therefore, code such as this:

```
Function PopulateChart

dim xmlString
xmlString = "<Chart><Series1>" _
          & "<Points><Point Label='Jan' Value='10' /></Points>" _
          & "<Points><Point Label='Feb' Value='20' /></Points>" _
          & "<Points><Point Label='Mar' Value='5' /></Points>" _
          & "<Points><Point Label='Apr' Value='15' /></Points>" _
          & "</Series1></Chart>"
Call CallSYSPROFunction(ChartPopulate, "ANDAPDC0", xmlString)

end function
```

will produce a chart like this:



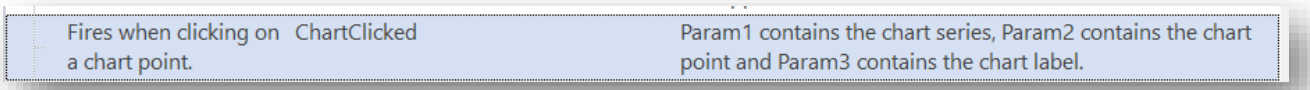
Hovering over a column will show a tooltip of the x axis and y axis values, and if there is a tooltip defined in the series that will be shown as well:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Events

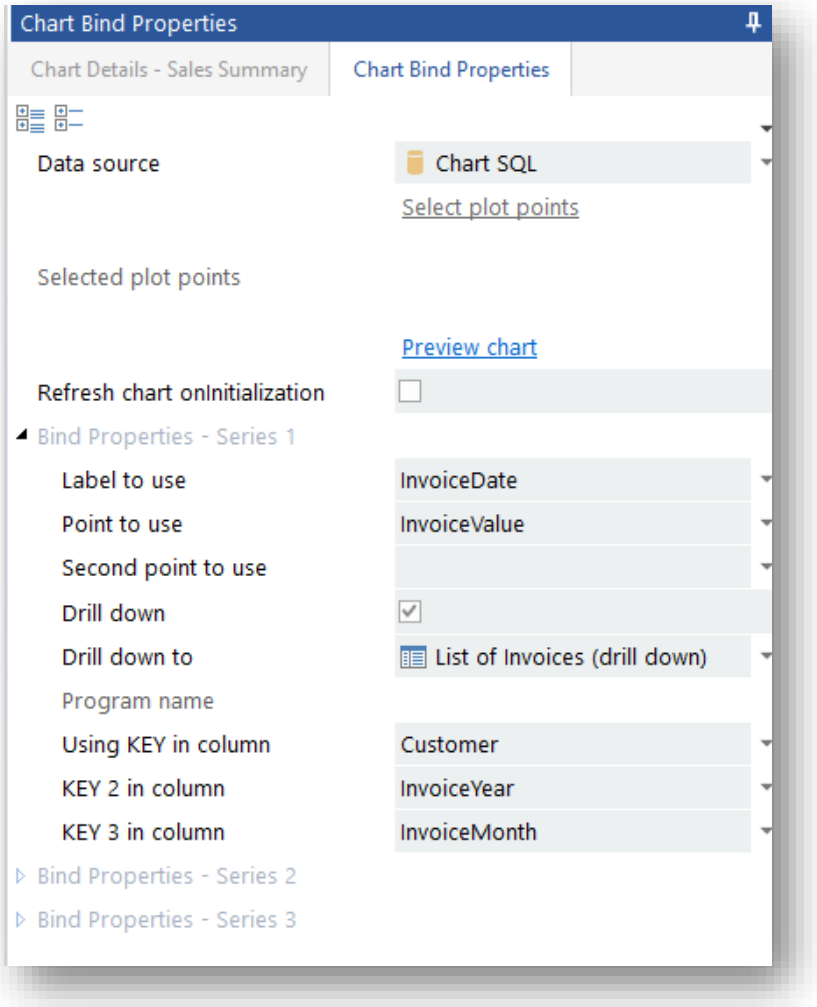
Clicking on a column raises the event **`ChartClicked`** event:



Data sources and Charts

You can bind data sources to any chart. Click on the **Chart Bind Properties** tab to see the available options and properties.

Typically, a chart's content will be refreshed when a specific data source is executed, either from being associated with an event (such as clicking on a toolbar button) or via code in a script. But you can also select the **Refresh chart OnInitialization** check box and the chart's data source will be executed as the application loads – note, however, that the data source would need to be constructed so that a 'key' is not required for it to execute correctly.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

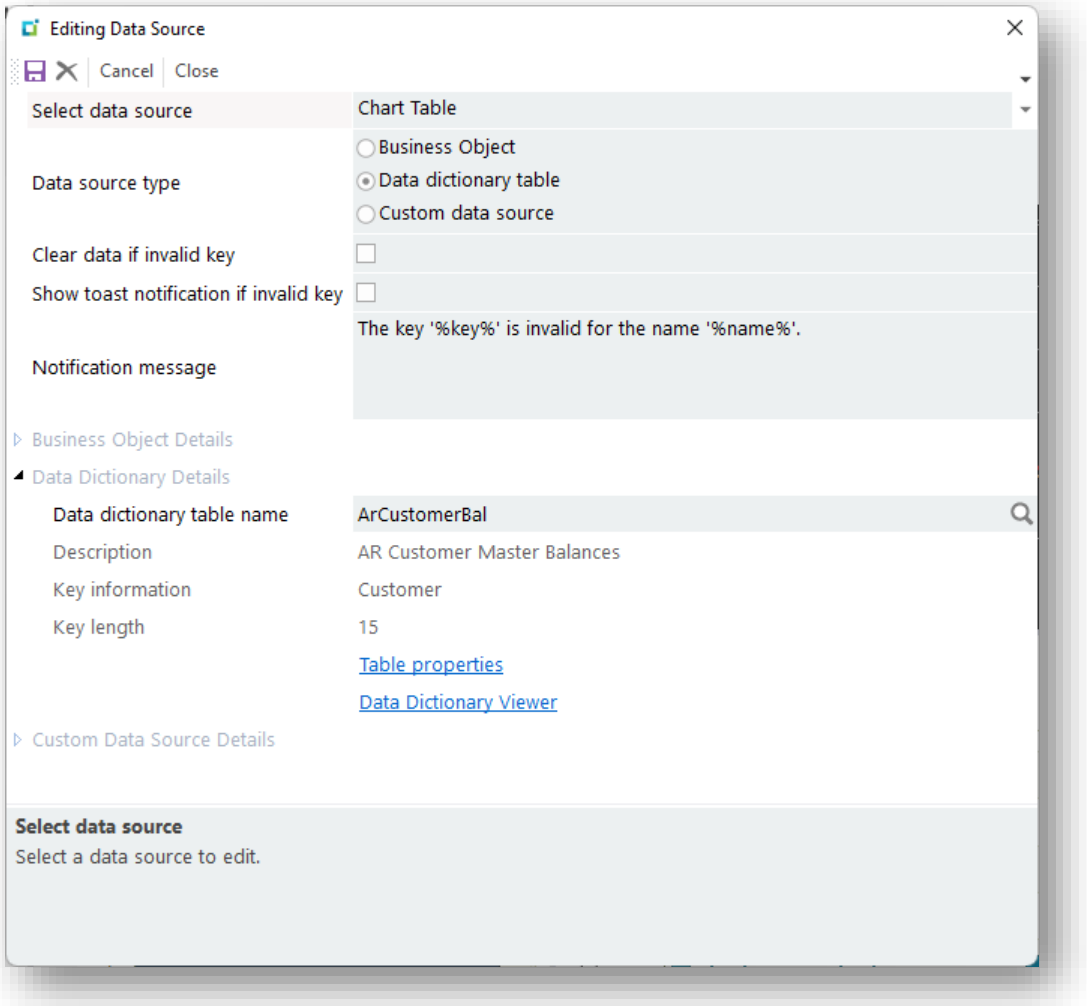
Select the data source to bind data for the chart. While only one data source can be selected per chart, you can select different elements from the data source to be shown in different series.

How data is bound to a chart depends on the type of data source.

Binding data to a chart using a Table data source

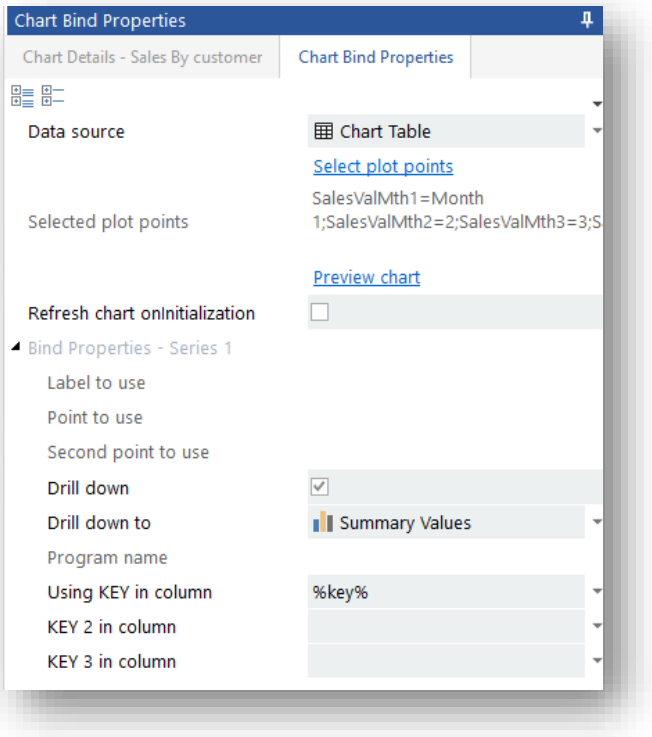
A Table data source can only provide chart points for the columns in the table.

Here's an example of a data source based on the **ArCustomerBal** table.

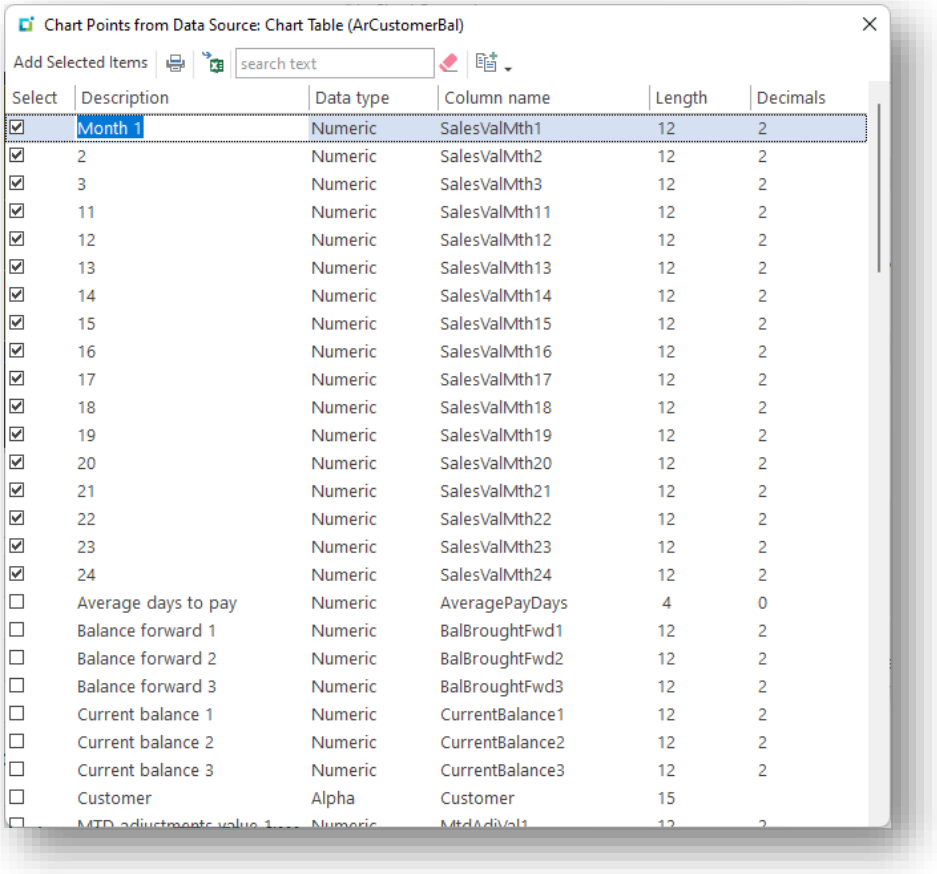


© SYSPRO. All Rights Reserved. All trademarks are recognized.

The Chart Bind properties will look like this:



Click on **Select plot points** and the following window will appear:



Select	Description	Data type	Column name	Length	Decimals
☑	Month 1	Numeric	SalesValMth1	12	2
☑	2	Numeric	SalesValMth2	12	2
☑	3	Numeric	SalesValMth3	12	2
☑	11	Numeric	SalesValMth11	12	2
☑	12	Numeric	SalesValMth12	12	2
☑	13	Numeric	SalesValMth13	12	2
☑	14	Numeric	SalesValMth14	12	2
☑	15	Numeric	SalesValMth15	12	2
☑	16	Numeric	SalesValMth16	12	2
☑	17	Numeric	SalesValMth17	12	2
☑	18	Numeric	SalesValMth18	12	2
☑	19	Numeric	SalesValMth19	12	2
☑	20	Numeric	SalesValMth20	12	2
☑	21	Numeric	SalesValMth21	12	2
☑	22	Numeric	SalesValMth22	12	2
☑	23	Numeric	SalesValMth23	12	2
☑	24	Numeric	SalesValMth24	12	2
☐	Average days to pay	Numeric	AveragePayDays	4	0
☐	Balance forward 1	Numeric	BalBroughtFwd1	12	2
☐	Balance forward 2	Numeric	BalBroughtFwd2	12	2
☐	Balance forward 3	Numeric	BalBroughtFwd3	12	2
☐	Current balance 1	Numeric	CurrentBalance1	12	2
☐	Current balance 2	Numeric	CurrentBalance2	12	2
☐	Current balance 3	Numeric	CurrentBalance3	12	2
☐	Customer	Alpha	Customer	15	

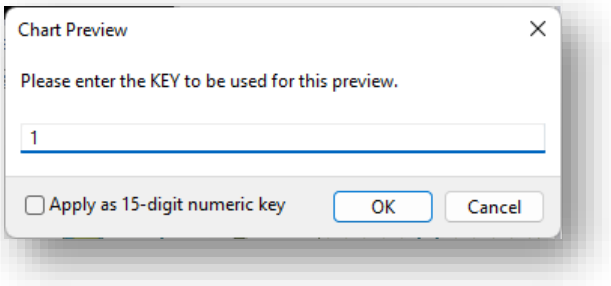
The chart points are displayed for the selected items and in the sequence from left to right (as presented in the list).

© SYSPRO. All Rights Reserved. All trademarks are recognized.

First, you would select the columns that you want plotted and then drag the item up or down the list so that they are displayed in the correct sequence. You can drag more than one item by selecting multiple rows and then drag them up or down, and then click the **Select** check box while holding down the **CTRL** key.

Finally, you can also change the description of each item that will be shown as the chart label – just click in the **Description** column and type in the required new label.

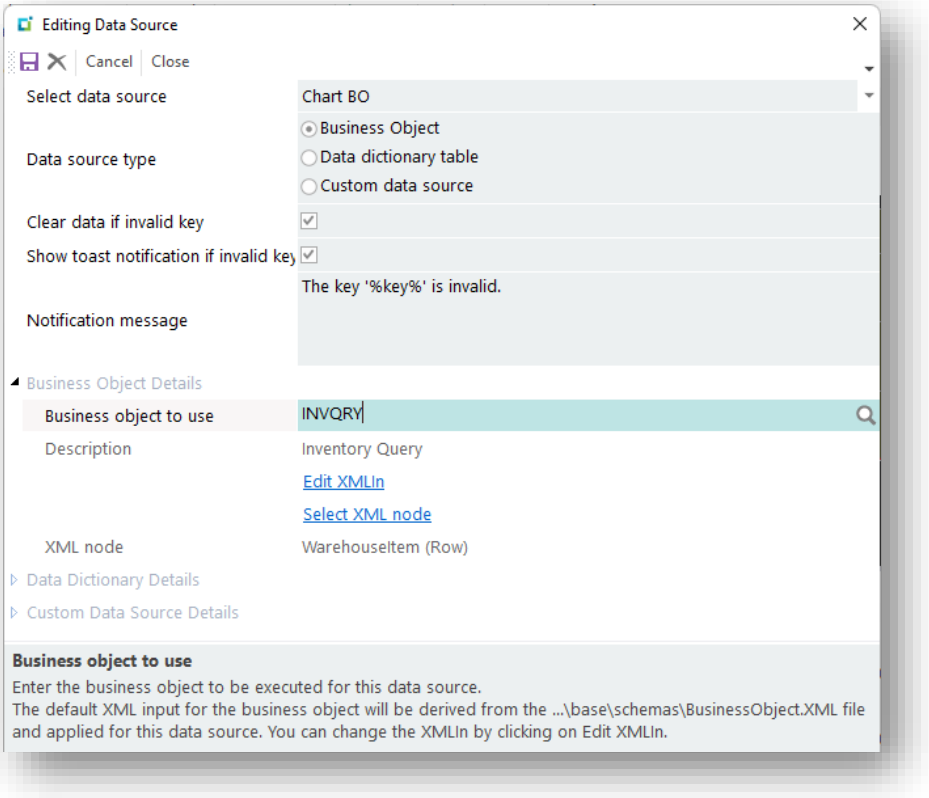
You can test out if your chart will work as expected by clicking on the **Preview chart** function – you will be prompted to enter a KEY for the preview to work:



Binding data to a chart using a Business Object data source

A business object data source can provide chart points for any columns provided from the selected XML node in the returned data.

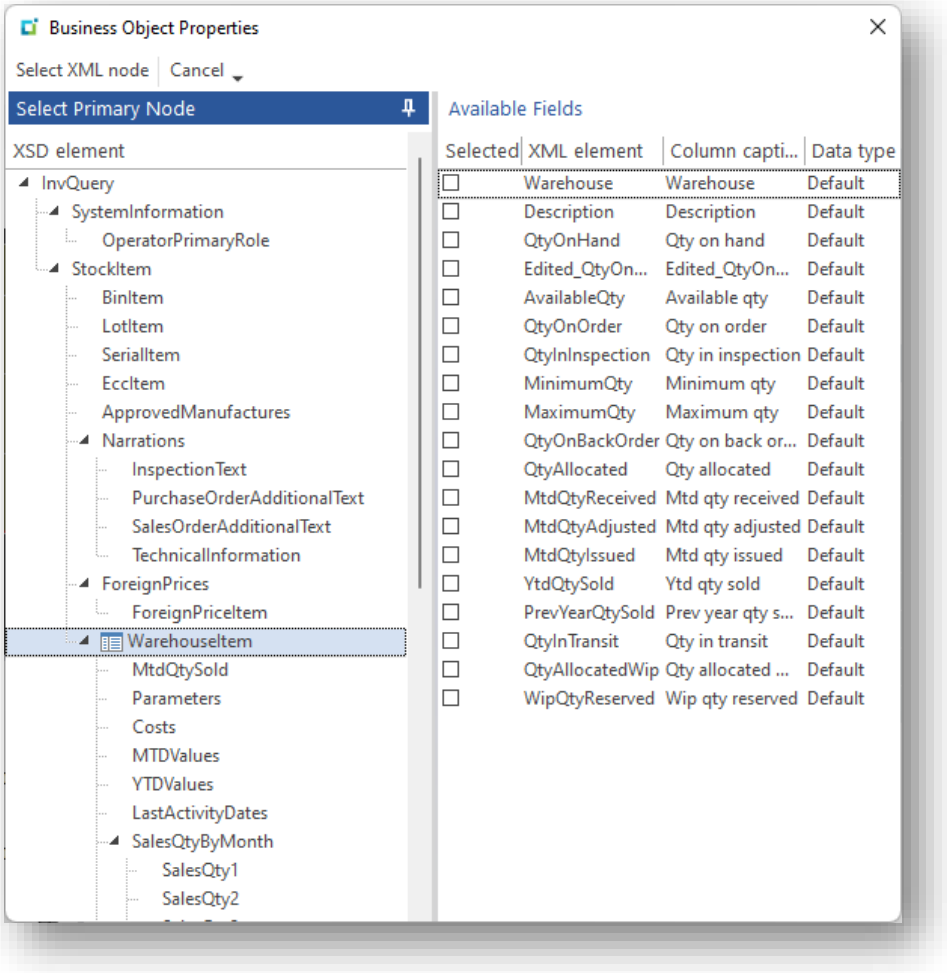
In the following example, INVQRY is used to show all the warehouses in a chart:



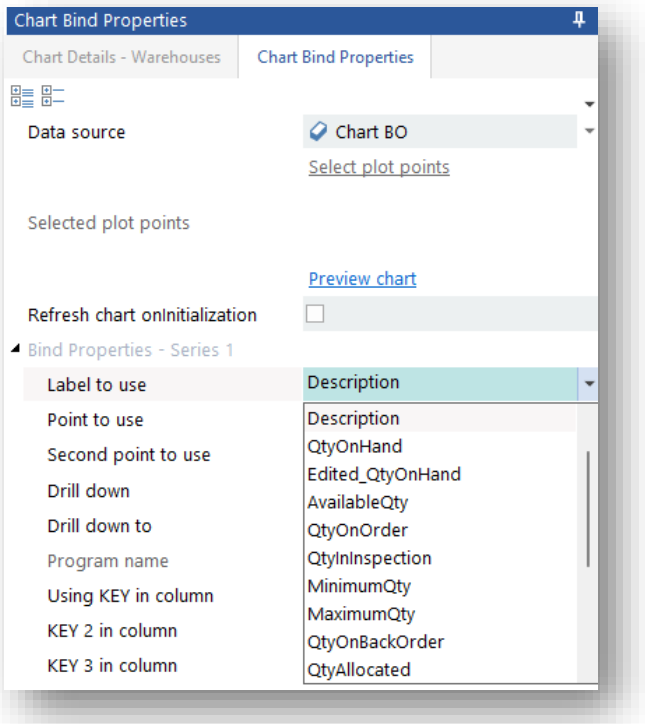
Clicking on **Select XML node** allows you to select which XML node element to process.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

In this case the **WarehouseItem** node is selected as this is a repeating row:



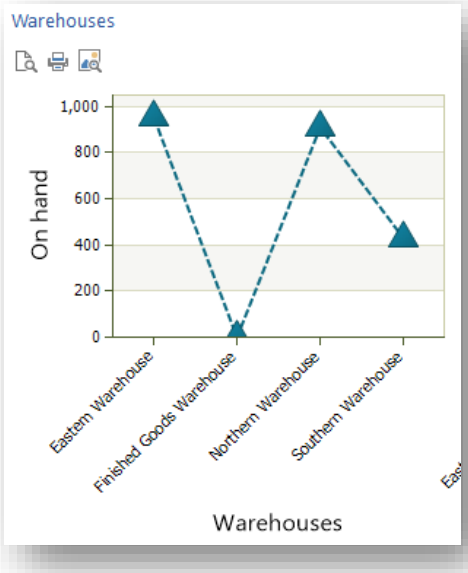
Within the Bind properties for the chart, the Business Object data source can be selected:



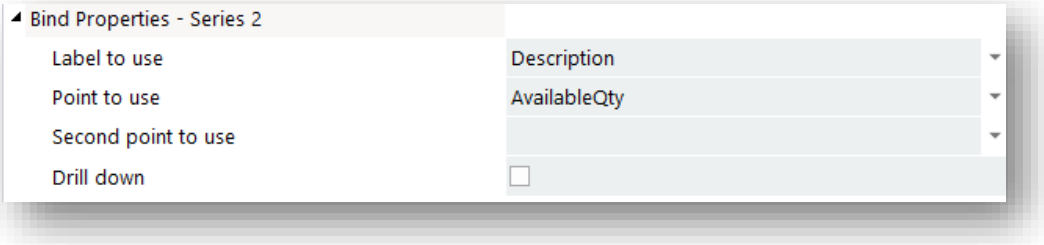
© SYSPRO. All Rights Reserved. All trademarks are recognized.

You can now select any available XML element from the **WarehouseItem** XML node for the **LABEL**, and then separately indicate which element value to show as the **POINT TO USE**. The values derived from these elements are used as the plot labels and points.

Clicking on the **Preview** hyperlink will show this:



You can of course select different elements to show in another series. So, while series 1 shows the Warehouse description and Qty on Hand, Series 2 can show the Warehouse description and Available Qty:



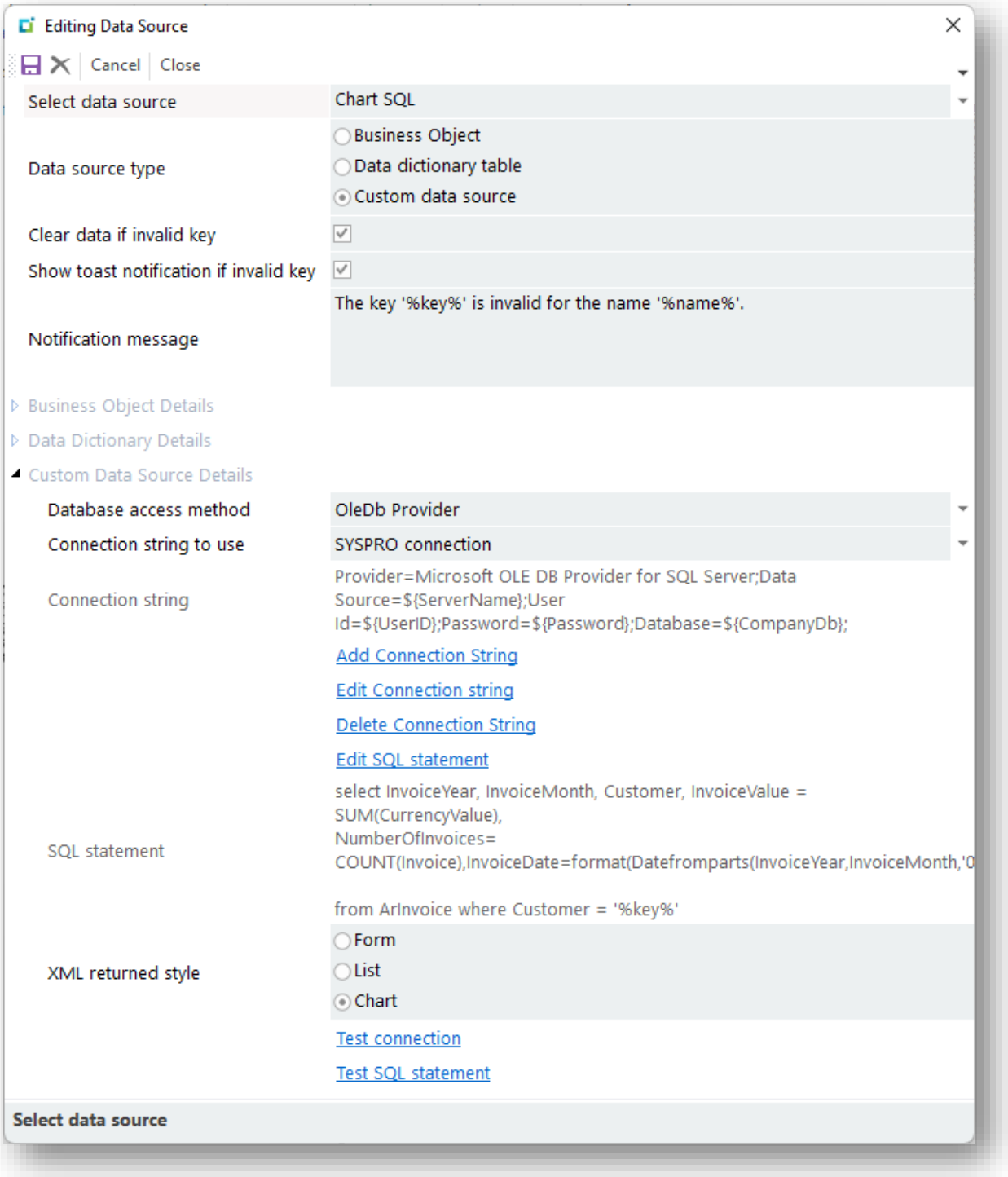
The result is this, assuming two diagrams and each series is in a different diagram:

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Binding data to a chart using a Custom data source

A custom data source can provide any data from any database. What's important to understand is that the chart can only plot points that are returned from the **SELECT** items in the SQL statement being executed.

Here's an example of a data source that aggregates invoice values for a given customer:



You will notice that the XML returned style is flagged as **Chart**. This ensures that the SQL statement will return Labels and Points in the XML string.

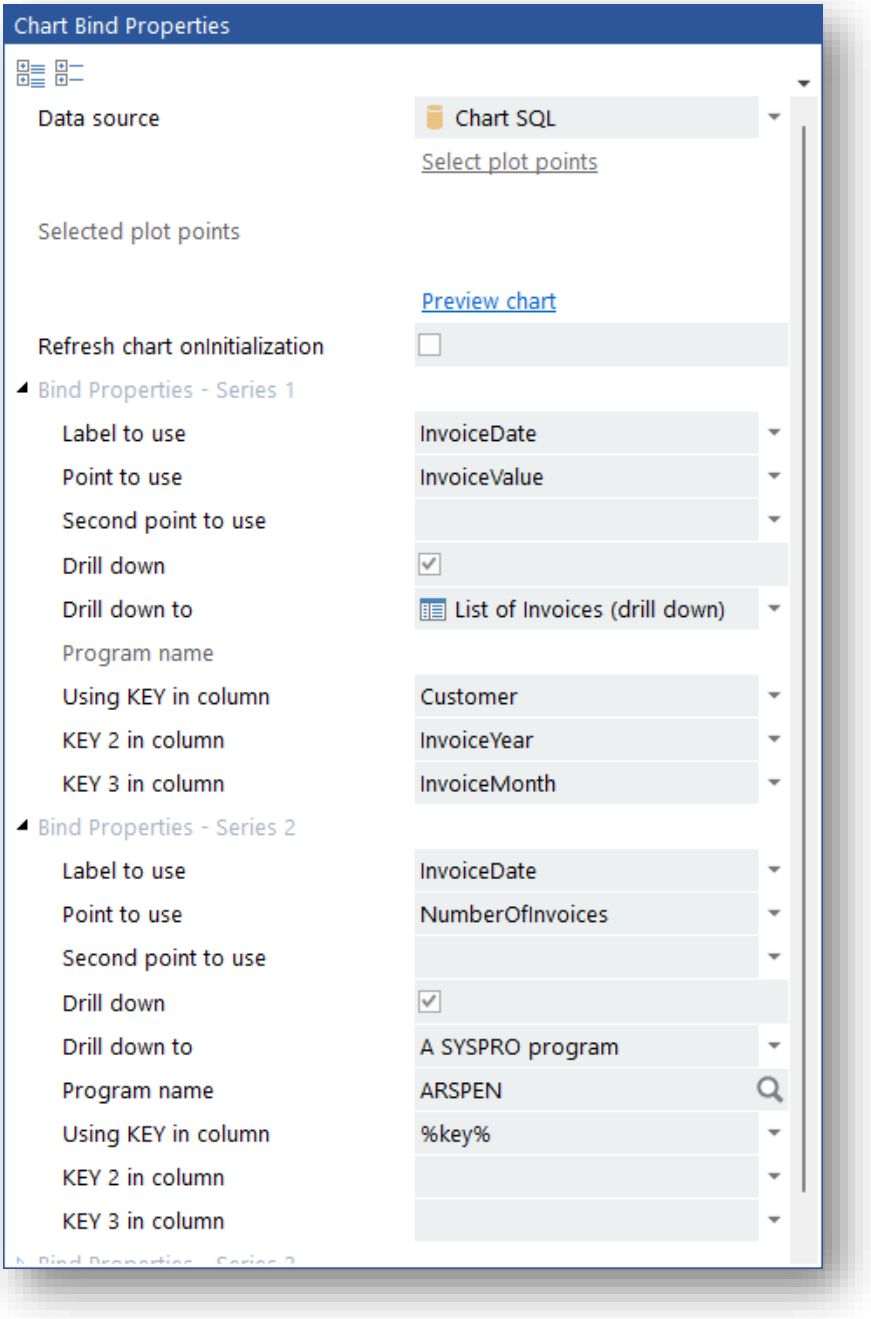
The SQL statement for this data source looks like this:

© SYSPRO. All Rights Reserved. All trademarks are recognized.

```
select InvoiceYear, InvoiceMonth, Customer, InvoiceValue = SUM(CurrencyValue),
NumberOfInvoices=
COUNT(Invoice),InvoiceDate=format(Datefromparts(InvoiceYear,InvoiceMonth,'01'),'yy
yy-MMM')
from ArInvoice where Customer = '%key%'
Group by InvoiceYear, InvoiceMonth, Customer
order by InvoiceYear, InvoiceMonth
```

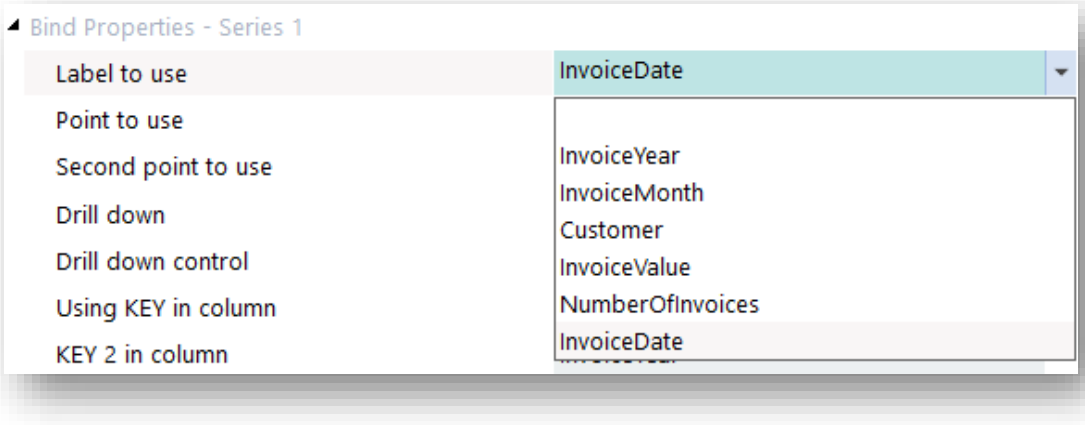
Notice the items in BOLD in the SELECT statement, as these will be the point values that can be selected to be displayed within the chart.

In the Bind properties for the chart, the SQL data source can be selected:

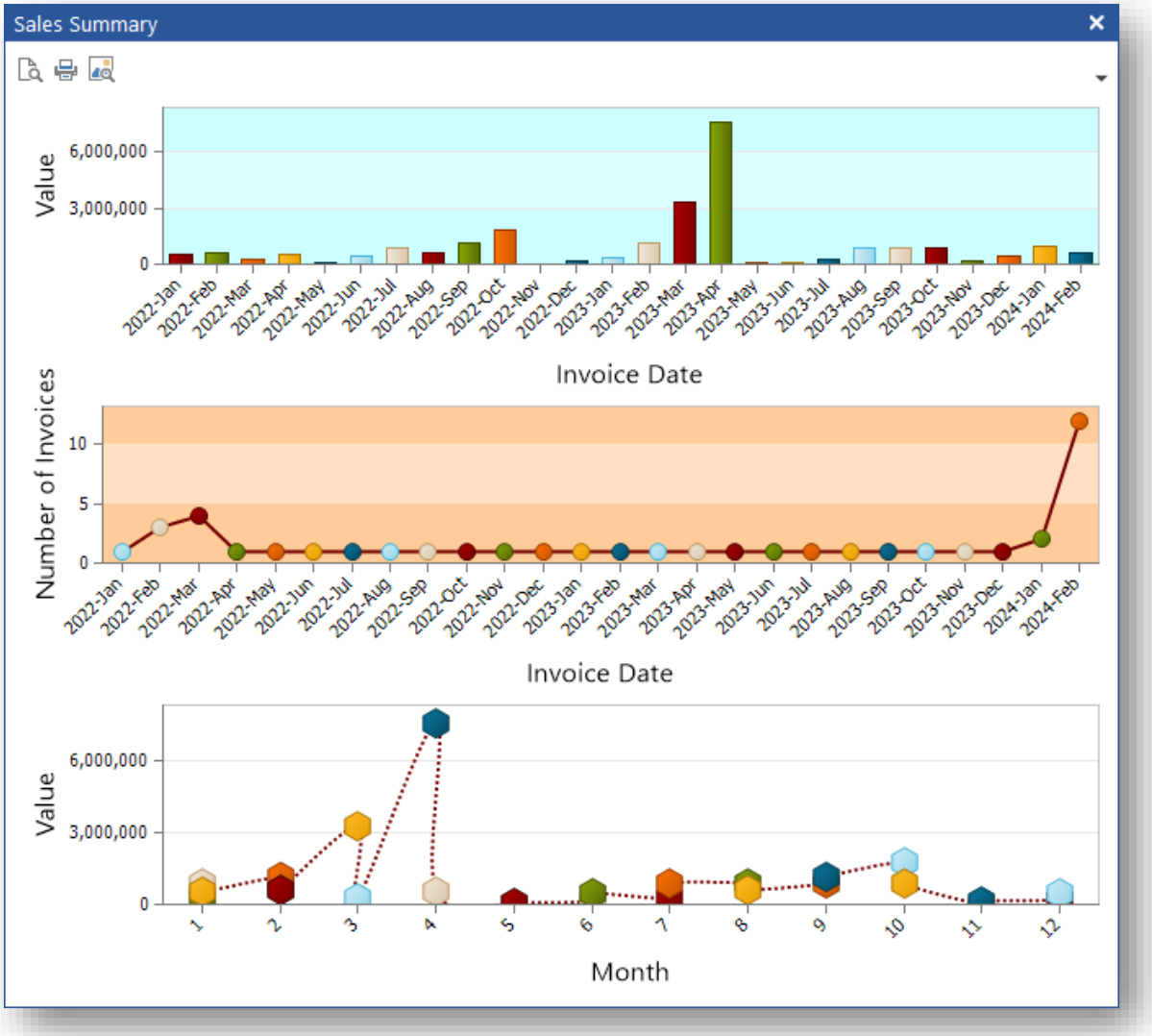


When you click on a LABEL or POINT TO USE dropdown list, you will see the available point values:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



When previewing the chart, it will look like this (each series is being shown in a different diagram):



© SYSPRO. All Rights Reserved. All trademarks are recognized.

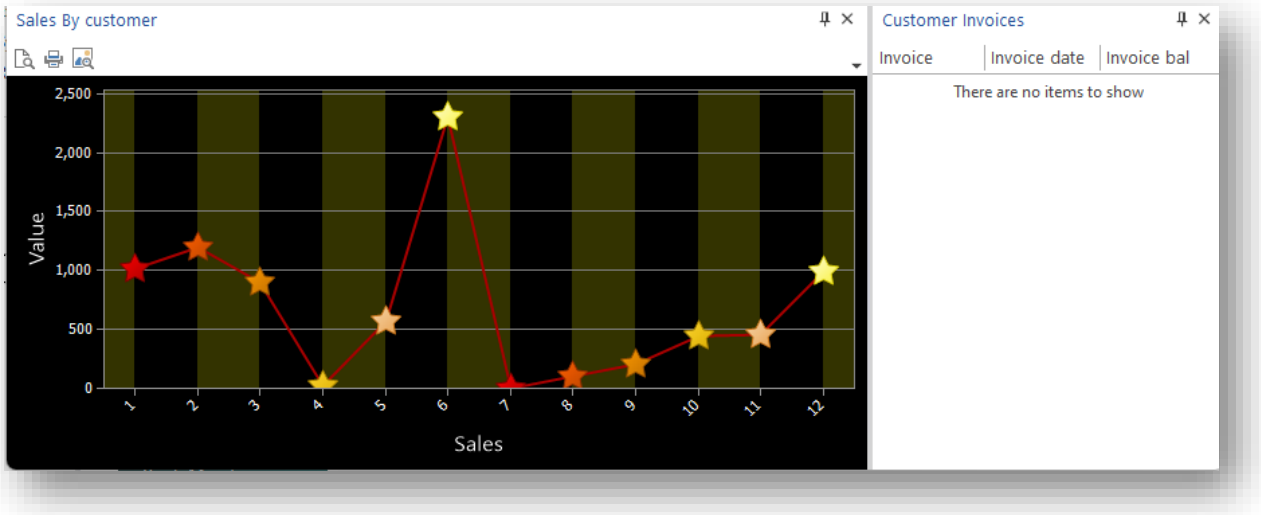
USING THE DRILL DOWN CAPABILITY FOR A CHART

You can, of course, use script to detect that a chart point has been clicked on and then provide your own drill down capabilities.

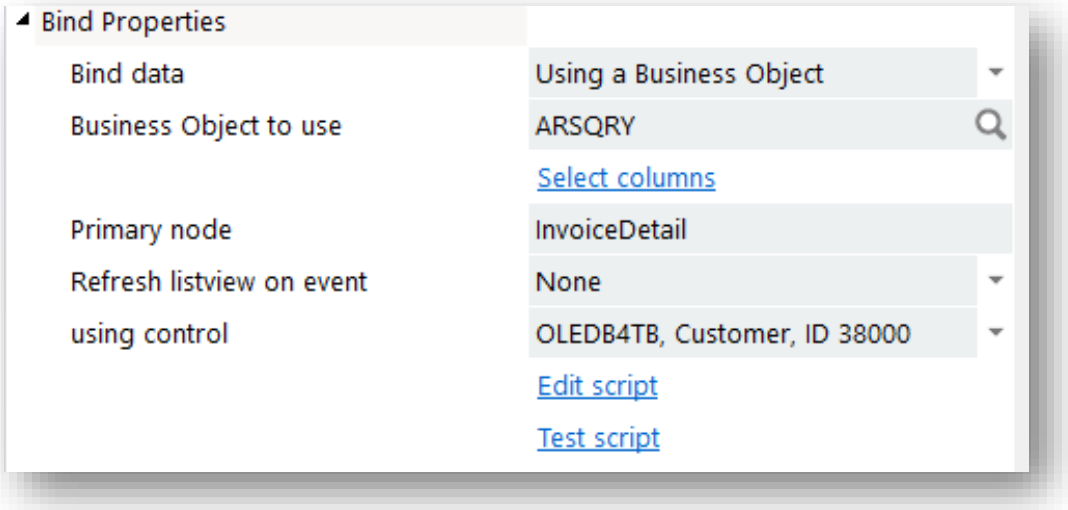
This section explains how you can use a data source to provide drill down capabilities without using any code.

Using our TABLE data source example, let's assume you want to see all the invoices for the customer that were used to execute the TABLE data source.

First, you will need to create a control to host a list view:

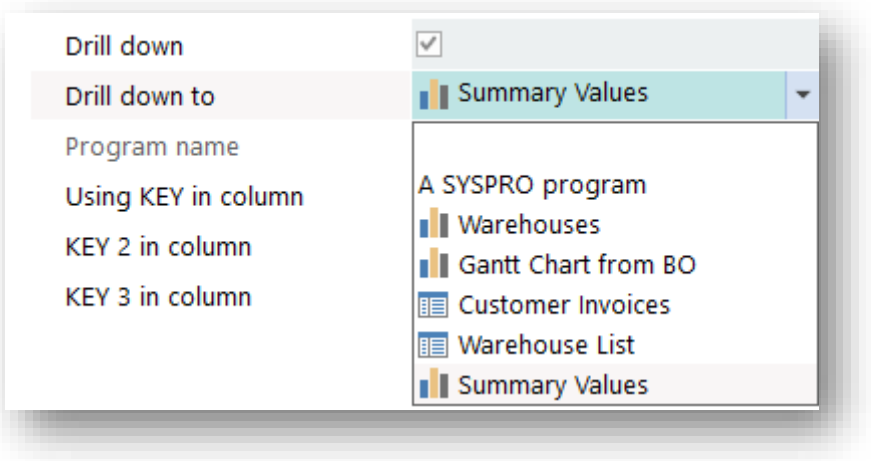


The BIND properties for the List view are simple enough, just using the ARSQRY business object and selecting the **InvoiceDetail** XML node to derive the list of invoices:



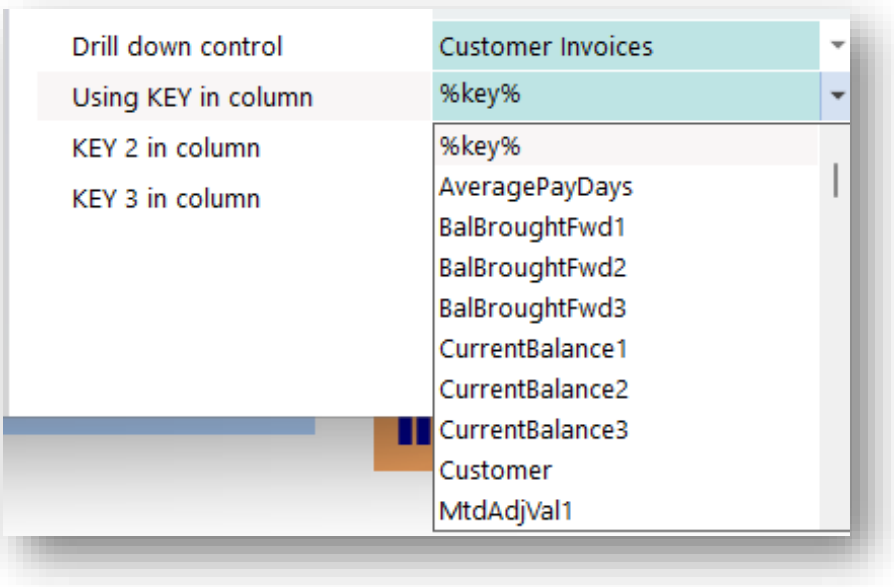
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Next, in the BIND properties for Series 1 for the chart, the **Drill down** option is enabled and the **Customer Invoices** control is selected from the **Drill down control** list:



Note: Only controls that are list views or charts will be shown in this list. Alternatively, you can select **A SYSPRO program** and then enter a program name to be executed for the drill down – the KEY value will be passed to the program.

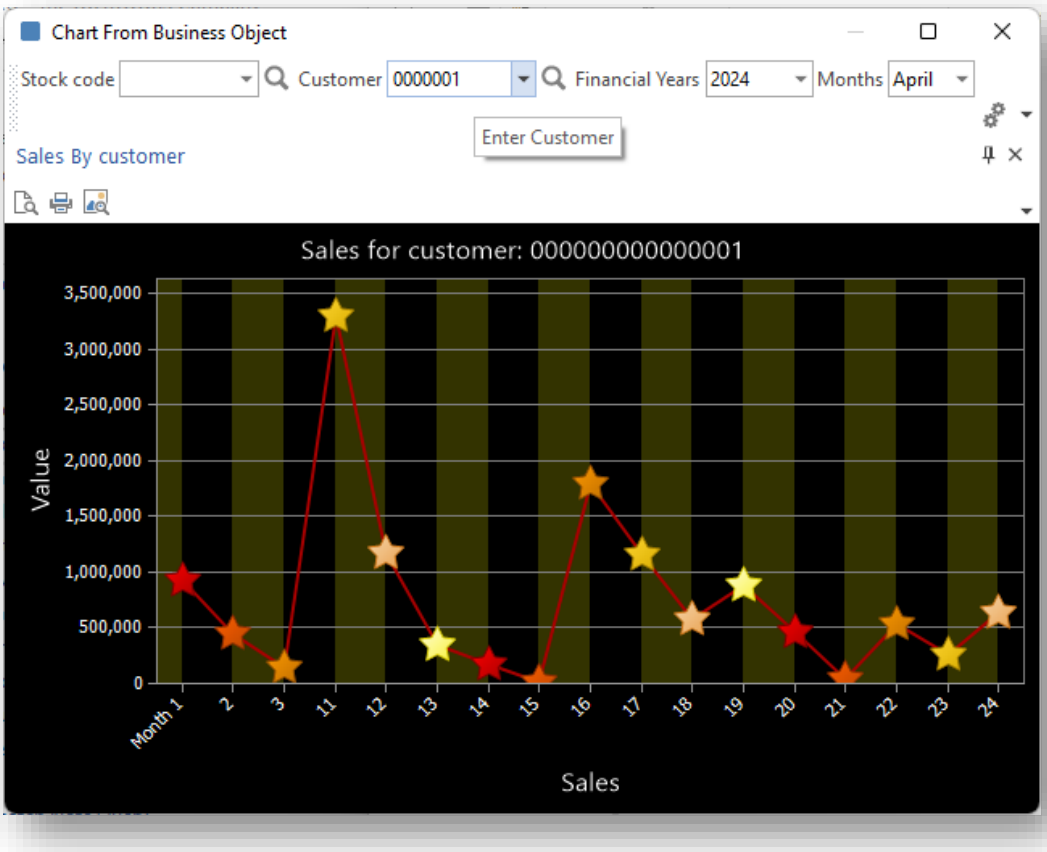
Finally, you can select the column in the TABLE that will provide the **KEY** to be passed to data source in the Customer Invoices control:



You could have selected **Customer** as the obvious KEY to be used, but you can also use the variable `%key%` - as this indicates that whatever key was used to execute the chart data source will be the same value passed to the drill down control. In our case, this is the Customer key.

When you run the application and enter a Customer number, the chart could look like this:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



Click on any data point and the invoice list will be displayed:

Customer Invoices

Invoice	Invoice date	Invoice bal
100100	2023-01-18	5.67
100466	2023-02-09	448000.00
100476	2023-03-05	914000.00
100497	2023-04-01	584800.00
100509	2023-04-08	2110.00
100510	2023-04-08	784.00
100511	2023-04-08	1190.10
100512	2023-04-08	16493.75
100513	2023-04-08	2041.20
100514	2023-04-09	2100.00
102100	2022-02-16	12.99
800055	2023-04-08	-86.00
900036	2023-04-08	250.00
_CRD01	2023-03-31	-2500.00

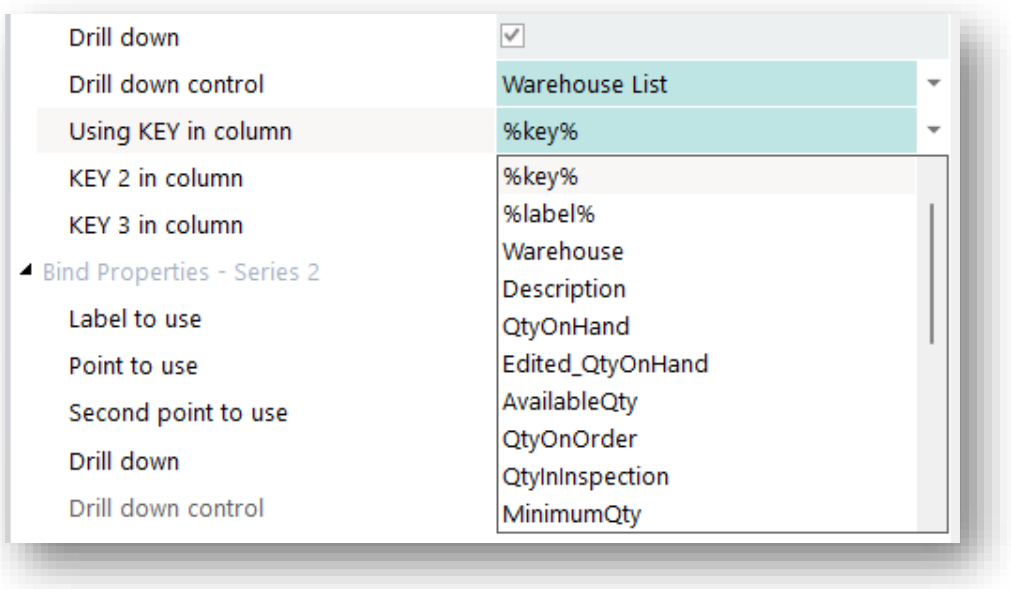
Note that the drill down control will be automatically made visible (if hidden or closed) or shown as a modal dialog box (if the pane property **Show pane as dialog box** is checked).

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Drill down from Business Object or Custom data source

The same principle applies to a business object or custom data source.

One difference is that you can optionally select the variable %label%:



The **`%label%`** variable could be useful if the label for each data point is the code or key to execute another data source. For example, instead of showing the DESCRIPTION as the label, **Warehouse** could have been selected and in which case the value of the warehouse can be passed as the key to the drill down control.

Another difference is that you can select up to 3 KEY values to be passed to the drill down data source – this is only of use where the drill down data source is a Custom data source that can use these extra key values to execute a SQL statement. The 3 KEY values will be inserted into placeholders **`%key%`**, **`%key2%`** and **`%key3%`** .

Here's an example of such a SQL statement:

```
select InvoiceYear, InvoiceMonth, CurrencyValue
from ArInvoice
where Customer = '%key%'
and InvoiceYear = '%key2%' and InvoiceMonth = '%key3%'
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

LIST VIEW CONTROL

SOME BASICS

- A List view is populated by data from an XML string.
- The XML string must have a root node.
- Repeating rows of data should be contained in a repeating node.
- The List view control calls this repeating node “Primary node”.
- The data that populates a column is held in the text of child nodes of the repeating/primary node.
- When populating the data in a row, the List view control matches element/node names of the XML string with the column headings in the List view.

For example:

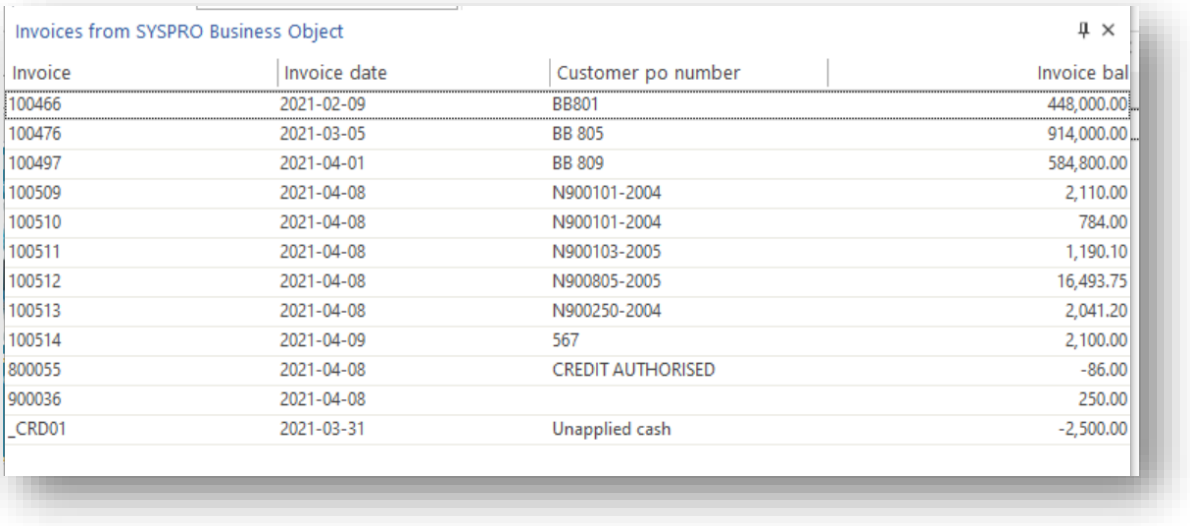
```
<Customer>
    <InvoiceDetail>
        <Customer>0000001</Customer>
        <Invoice>100466</Invoice>
        <InvoiceDate>2021-02-09</InvoiceDate>
        <CustomerPoNumber>BB801</CustomerPoNumber>
        <InvoiceBal>448,000.00</InvoiceBal>
    </InvoiceDetail>
    <InvoiceDetail>
        <Customer>0000001</Customer>
        <Invoice>100476</Invoice>
        <InvoiceDate>2021-03-05</InvoiceDate>
        <CustomerPoNumber>BB 805</CustomerPoNumber>
        <InvoiceBal>914,000.00</InvoiceBal>
    </InvoiceDetail>
    <InvoiceDetail>
        <Customer>0000001</Customer>
        <Invoice>100497</Invoice>
        <InvoiceDate>2021-04-01</InvoiceDate>
        <CustomerPoNumber>BB 809</CustomerPoNumber>
        <InvoiceBal>584,800.00</InvoiceBal>
    </InvoiceDetail>
… etc …
</Customer>
```

In this example:

- The root node is **<Customer>**
- The repeating row node (Primary node) is **<InvoiceDetail>**
- The data comes from **<InvoiceDetail>** children
  (E.g. <CustomerPoNumber>BB801</CustomerPoNumber>)

© SYSPRO. All Rights Reserved. All trademarks are recognized.

The XML above produces a List view like this:



Invoice	Invoice date	Customer po number	Invoice bal
100466	2021-02-09	BB801	448,000.00
100476	2021-03-05	BB 805	914,000.00
100497	2021-04-01	BB 809	584,800.00
100509	2021-04-08	N900101-2004	2,110.00
100510	2021-04-08	N900101-2004	784.00
100511	2021-04-08	N900103-2005	1,190.10
100512	2021-04-08	N900805-2005	16,493.75
100513	2021-04-08	N900250-2004	2,041.20
100514	2021-04-09	567	2,100.00
800055	2021-04-08	CREDIT AUTHORISED	-86.00
900036	2021-04-08		250.00
_CRD01	2021-03-31	Unapplied cash	-2,500.00

The XML string can contain far more data than the List view needs. The extra data is simply ignored.

For example – using the SYSPRO business object ARSQRY, the XML looks like this:

```
<?xml version="1.0" encoding="Windows-1252"?>
<ARStatement Language='05' Language2='EN' CssStyle='' DecFormat='1' DateFormat='01' Role='01'
Version='8.0.034' OperatorPrimaryRole='   '>
    <Header>
        <CompanyName>SYSPRO DEMONSTRATION</CompanyName>
        <CompAddress1 />
        <CompAddress2>P O Box 777</CompAddress2>
        <CompAddress3>Great Outdoors</CompAddress3>
        <CompTaxNo />
        <CompRegNo>1978/035534/04</CompRegNo>
        <Customer>0000001</Customer>
… etc …
 </Header>
    <InvoiceDetail>
        <DocumentType>I</DocumentType>
        <Invoice>100466</Invoice>
        <InvoiceDate>2021-02-09</InvoiceDate>
        <DueDate>2021-03-11</DueDate>
        <CustomerPoNumber>BB801</CustomerPoNumber>
        <CurrencyValue>       448000.00</CurrencyValue>
        <InvoiceTerms>0</InvoiceTerms>
        <TBLART_Description>30 Days - Net</TBLART_Description>
        <InvoiceBal>       448000.00</InvoiceBal>
        <MonthPosted>12</MonthPosted>
        <OriginalAmount>       448000.00</OriginalAmount>
        <Edited_OriginalAmount>448000.00</Edited_OriginalAmount>
        <Salesperson>100</Salesperson>
        <SalespersonName>Tony Dean</SalespersonName>
        <Area>N</Area>
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

```
        <AreaDesc>Northern Region</AreaDesc>
        <InvoiceAgeing>2022-04-02</InvoiceAgeing>
        <Branch>10</Branch>
        <SalesOrder>000793</SalesOrder>
        <DocumentFormat>0</DocumentFormat>
        <LastDelNote />
        <ProofOfDelivery>Y</ProofOfDelivery>
        <PodReference>D100273</PodReference>
        <PodEntryDate>2021-04-08</PodEntryDate>
        <StoreNumber />
        <Bucket>            7</Bucket>
        <DunningGroup />
        <DunningStatus>000000</DunningStatus>
        <DunningDate />
        <DunningInvoiceBal>0.00</DunningInvoiceBal>
        <DunningUser />
    </InvoiceDetail>
    <InvoiceDetail>
        <DocumentType>I</DocumentType>
        <Invoice>100476</Invoice>
        <InvoiceDate>2021-03-05</InvoiceDate>
        <DueDate>2021-04-04</DueDate>
        <CustomerPoNumber>BB 805</CustomerPoNumber>
        <CurrencyValue>       914000.00</CurrencyValue>
        <InvoiceTerms>0</InvoiceTerms>
        <TBLART_Description>30 Days - Net</TBLART_Description>
        <InvoiceBal>       914000.00</InvoiceBal>
        <MonthPosted>01</MonthPosted>
        <OriginalAmount>       914000.00</OriginalAmount>
        <Edited_OriginalAmount>914000.00</Edited_OriginalAmount>
        <Salesperson>100</Salesperson>
        <SalespersonName>Tony Dean</SalespersonName>
        <Area>N</Area>
        <AreaDesc>Northern Region</AreaDesc>
        <InvoiceAgeing>2022-04-02</InvoiceAgeing>
… etc …
```

In this example:

- The root node is **<ARStatement>**
- The repeating row node is **<InvoiceDetail>**

Any data in child nodes that do not match the column headings is ignored. So, this XML will produce the same list view as the previous example, despite it holding much more data.

The sample App SAMDTA is a good example of how different methods of retrieving data can create an XML string and use the same list view for the different data sources.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Considerations

- The Primary node, column headings and XML node names are case sensitive.
- If the List view does not populate at all (i.e. it remains blank) then it is likely that it cannot find its primary node.
  Check spelling and case sensitivity.
- If a column is missing data, then it means the list view cannot match the heading to any node in the xml.
  Check spelling and case sensitivity.

Advanced Formatting

The XML string can be a set of simple XML elements with values but can also be constructed with various properties such as making a cell editable (or not) and changing cell colors.

The various properties that can be assigned to a cell value include:

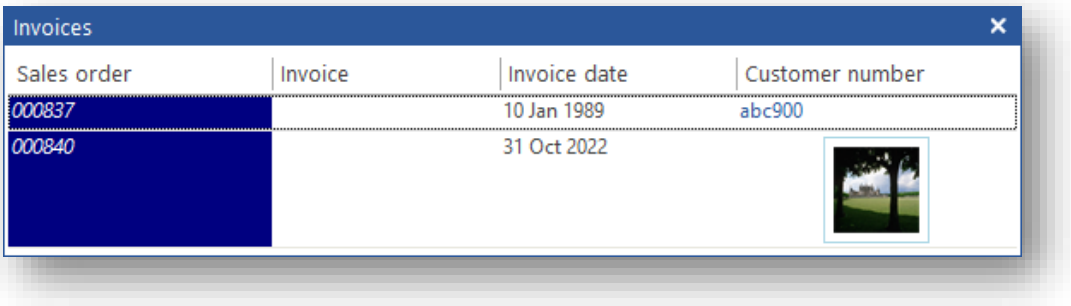
Property	Entry
IsReadOnly	True or False
Value	The value of the cell
Background	A named color
Foreground	A named color
Tooltip	The tooltip for the cell
IsBold	True or False
IsItalic	True or False
XAMLCode	A named XAML theme to be applied to the cell

Here is some example code:

```
   dim xmlString
   xmlstring = "<Customers><Customer>" &_
    "<InvoiceDetail>" &_
    "<SalesOrder IsReadOnly='true' Value='837' Background='Blue'
Foreground='white' IsItalic='true'/>" &_
    "<CustomerPoNumber IsReadOnly='true' Value='abc900' Tooltip='This is a po
number' />" &_
    "<InvoiceDate  Value='19890110'  />" &_
    "</InvoiceDetail>" &_
        "<InvoiceDetail>" &_
    "<SalesOrder IsReadOnly='true' Value='840' Background='Blue'
Foreground='white' IsItalic='true'/>" &_
    "<CustomerPoNumber IsReadOnly='true' Value='c:\images\chateau.jpg'
XAMLCode='Thumbnail' />" &_
        "<InvoiceDate  Value='20221031'  />" &_
    "</InvoiceDetail>" &_
    "</Customer></Customers>"
   call CallSYSPROFunction(ListviewClearAddRecords, "DTS004L0", xmlString)
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

This code will render as follows:



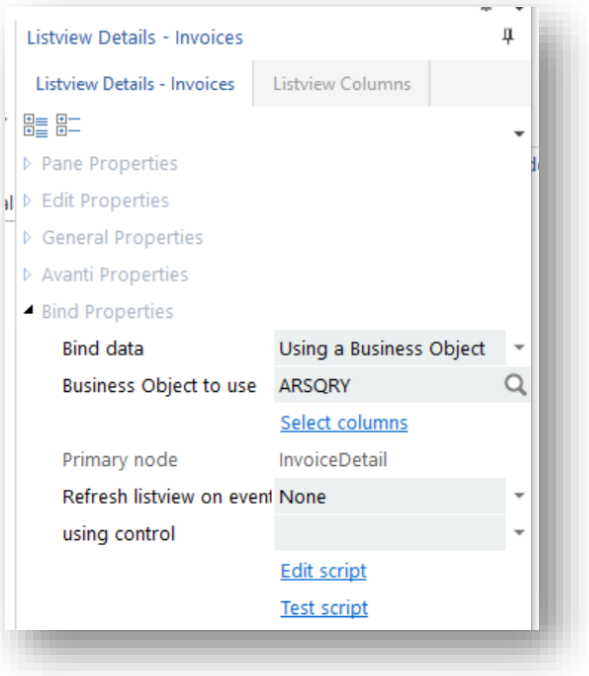
Note: Named colors may be:

- Black
- Blue
- Cyan
- White
- Green
- Red
- Magenta
- Yellow
- Gray

These may also be preceded by the word **Dark** or **Light** (e.g. Dark Cyan or Light Red).

SETTING UP THE LIST VIEW CONTROL

Designing the list view and binding it to a data source is done in the **Bind Properties** of the List view:

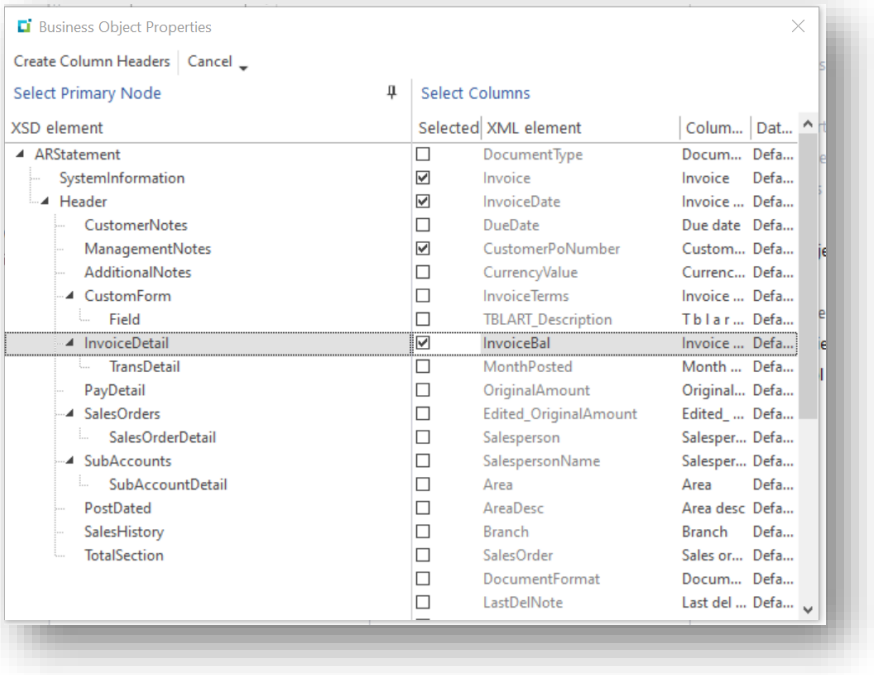


The key tasks to perform here are to set the “Primary node” and set the column headings.

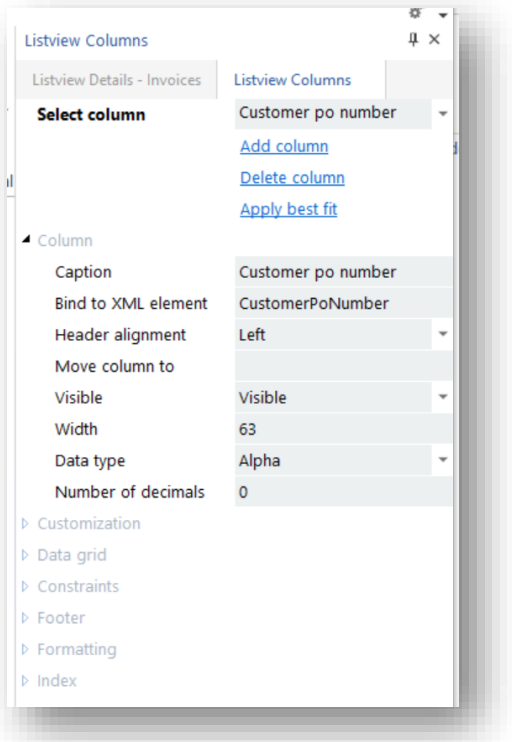
© SYSPRO. All Rights Reserved. All trademarks are recognized.

The **Bind data** option and the **Business Object to Use** help the user do this. But note, they have nothing to do with the actual operation of the list view when the App runs. They are used for design and configuration purposes only.

If **Bind data** is set to **Using a Business Object** and the **Business Object to use** is set, then the **Select Columns** option can be clicked: this launches a wizard that helps identify the Primary node and the columns required.



Columns can be added, deleted, and have properties set by selecting the **Listview Columns** pane.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Note that the column heading can have a different heading to the **Binding to the XML** element. In this case the list view will be looking for an XML element called "`CustomerPoNumber`" whilst the heading will display "Customer po number".

If the **Bind data** is set to **Using a custom data source**, then you must manually enter a **Primary node** and use the **Listview Columns** pane to create the columns.

Sending the XML string to the List view

Irrespective of the way the XML string is derived, sending the data to the list view in the VBScript is the same.

For example:

```
call CallSYSPROFunction(ListviewClearAddRecords, "SAMDTAL0", xmldata)
```

where the `xmldata` variable holds the XML string.

This callout function will clear the list view grid and pass the xml document to the list view.

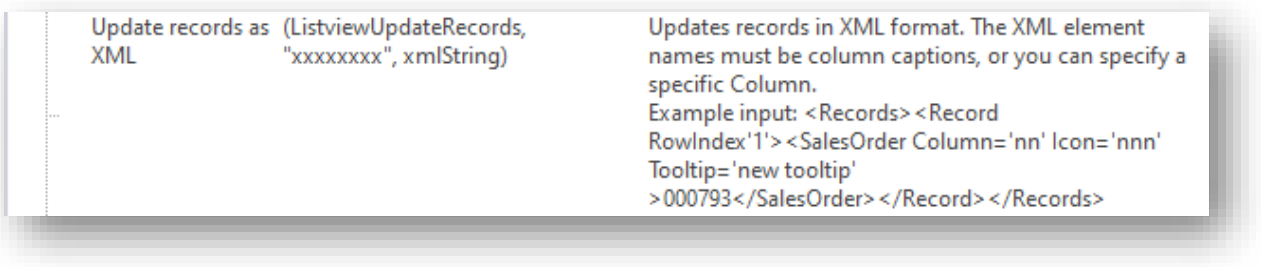
The list view identifies its primary node and matches the data elements with the column headings and displays the data.

You can update an existing Listview using the callout `ListviewUpdateRecords`.

This callout updates specific records in the grid using XML format. The XML string must contain the **RowIndex** (the row to update in the grid) and may contain new values, icons and/or tooltips.

The XML element names are used to match to the grid's column captions (for example, an XML name of **SalesOrder** will be used to update a column whose caption is **Sales order**).

If this is not precise enough for matching, you can instead provide the specific column index using the attribute **Column='nnn'**.



The XML String looks like this:

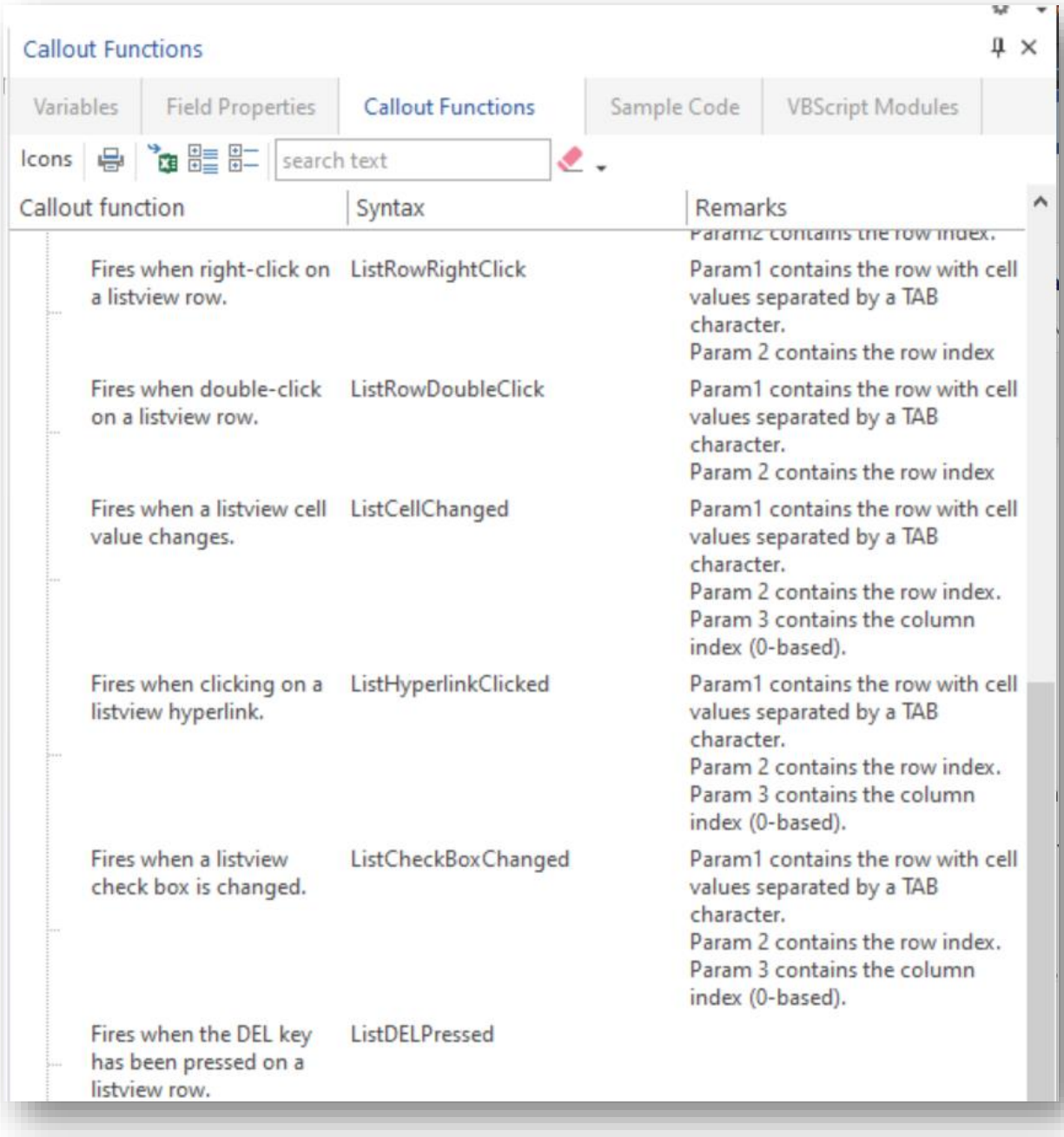
```
<Records><Record RowIndex='1'><Invoice Icon='223' Tooltip='Important!'
Column='1'>123456789</Invoice></Record></Records>
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

Manipulating the List View in the Designer

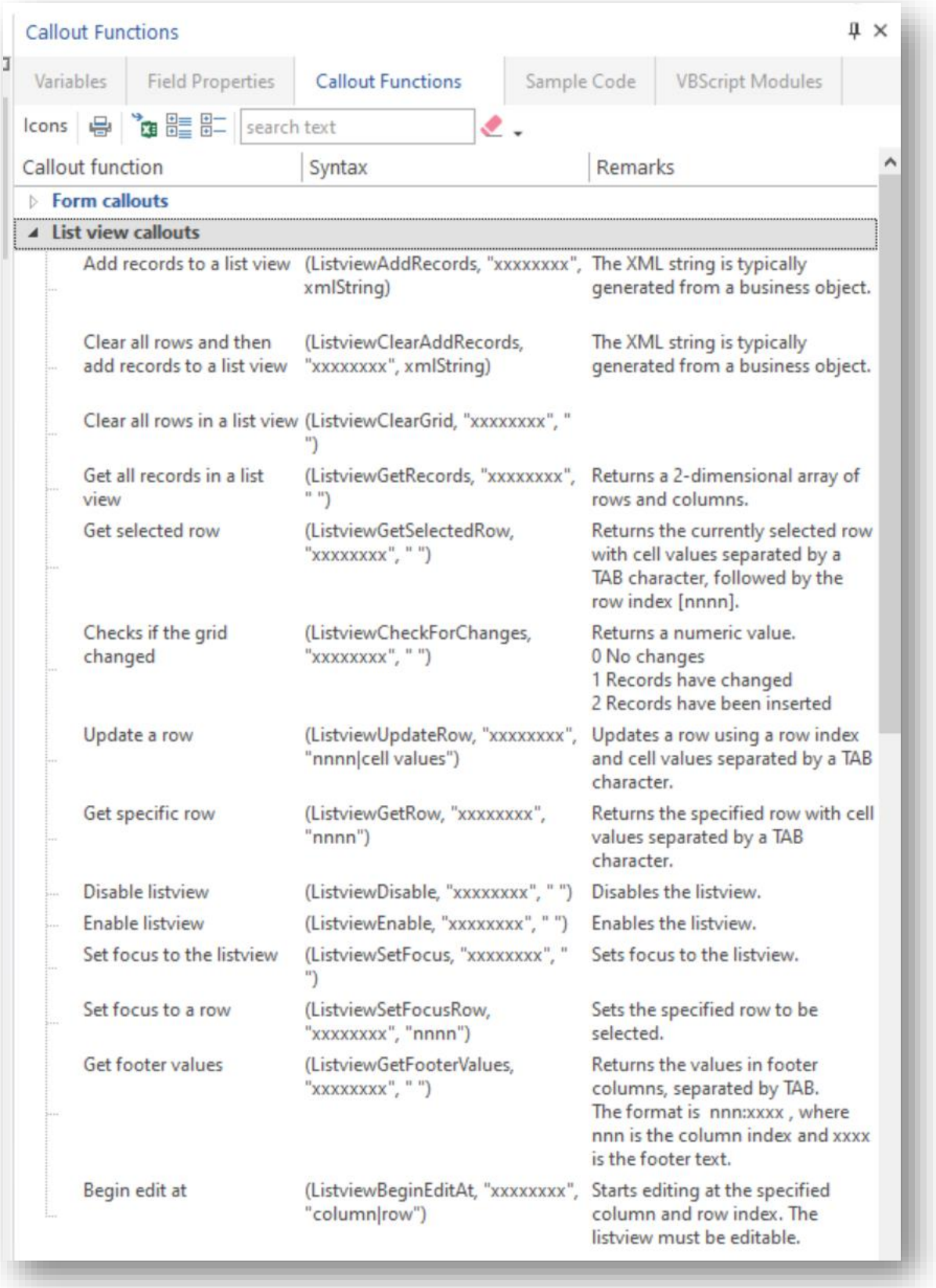
The list view control has several **Events** that can be captured by the VBScript.

These are indicated here:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

It also has several Callout Functions:



Callout function	Syntax	Remarks
Form callouts		
List view callouts		
Add records to a list view	(ListviewAddRecords, "xxxxxxxx", xmlString)	The XML string is typically generated from a business object.
Clear all rows and then add records to a list view	(ListviewClearAddRecords, "xxxxxxxx", xmlString)	The XML string is typically generated from a business object.
Clear all rows in a list view	(ListviewClearGrid, "xxxxxxxx", " ")	
Get all records in a list view	(ListviewGetRecords, "xxxxxxxx", " ")	Returns a 2-dimensional array of rows and columns.
Get selected row	(ListviewGetSelectedRow, "xxxxxxxx", " ")	Returns the currently selected row with cell values separated by a TAB character, followed by the row index [nnnn].
Checks if the grid changed	(ListviewCheckForChanges, "xxxxxxxx", " ")	Returns a numeric value. 0 No changes 1 Records have changed 2 Records have been inserted
Update a row	(ListviewUpdateRow, "xxxxxxxx", "nnnn|cell values")	Updates a row using a row index and cell values separated by a TAB character.
Get specific row	(ListviewGetRow, "xxxxxxxx", "nnnn")	Returns the specified row with cell values separated by a TAB character.
Disable listview	(ListviewDisable, "xxxxxxxx", " ")	Disables the listview.
Enable listview	(ListviewEnable, "xxxxxxxx", " ")	Enables the listview.
Set focus to the listview	(ListviewSetFocus, "xxxxxxxx", " ")	Sets focus to the listview.
Set focus to a row	(ListviewSetFocusRow, "xxxxxxxx", "nnnn")	Sets the specified row to be selected.
Get footer values	(ListviewGetFooterValues, "xxxxxxxx", " ")	Returns the values in footer columns, separated by TAB. The format is nnn:xxxx , where nnn is the column index and xxxx is the footer text.
Begin edit at	(ListviewBeginEditAt, "xxxxxxxx", "column|row")	Starts editing at the specified column and row index. The listview must be editable.

Using a combination of events and callouts it is possible to add, delete, and update rows within the list view.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

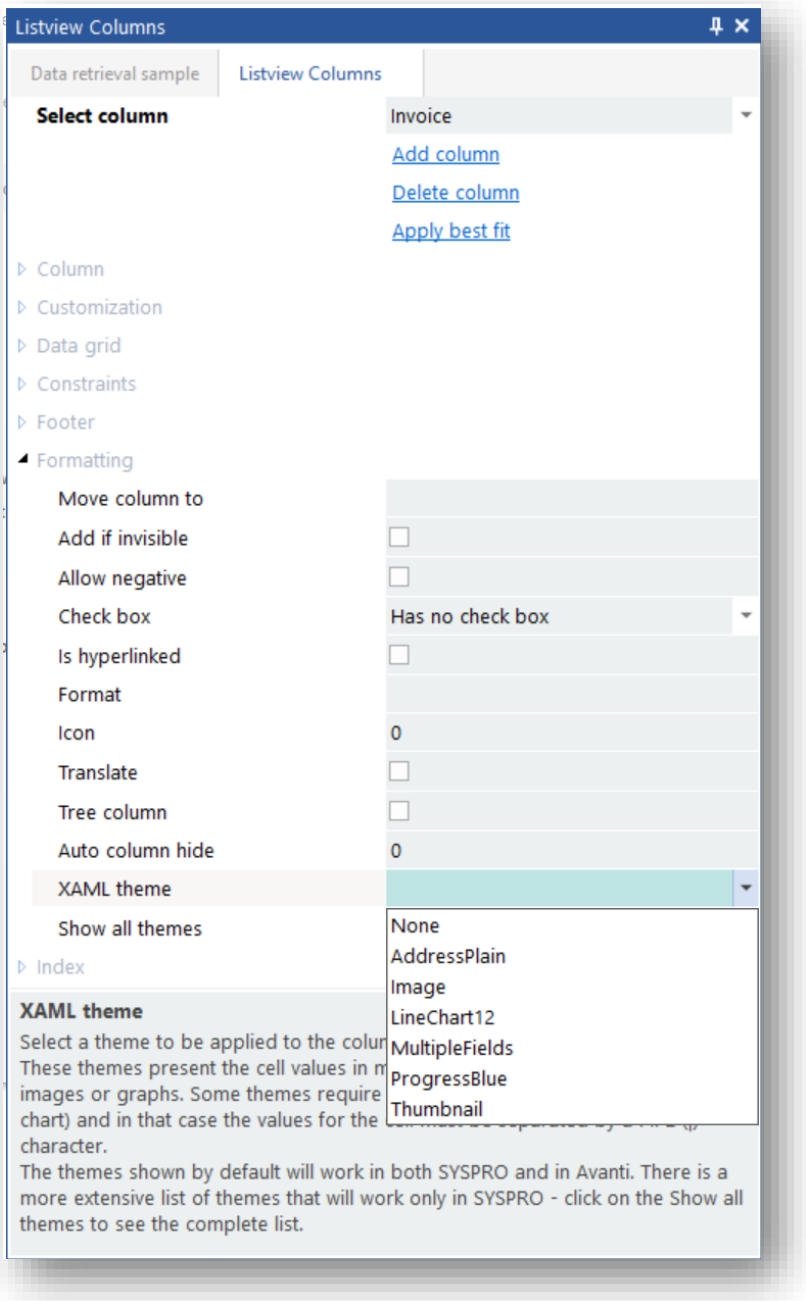
LIST VIEW COLUMN AND XAML

The data within a list view column can be rendered by a standard SYSPRO XAML theme.

This is done by selecting a XAML theme in the formatting section of the column definition.

For example:

- **Image** theme will render an image URL in the cell.
- **ProgressBlue** theme takes a number between 1 and 100 and shows a progress bar.
- **MultipleFields** takes a coma separated list of up to 9 pieces of data and displays each one on a different line.



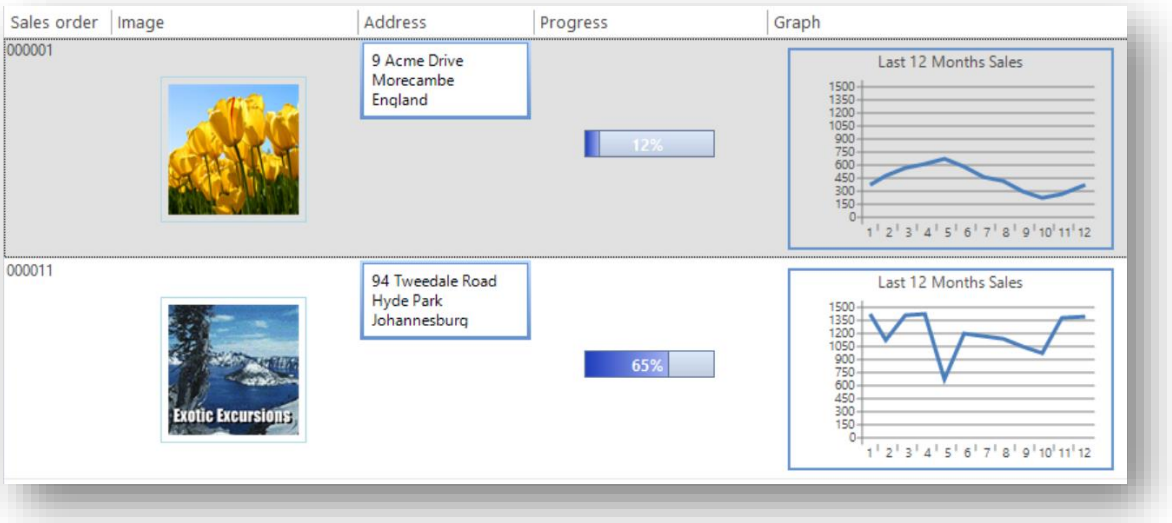
These themes are held in `base/Samples` and are named `Form_?????.XAML` (e.g. `Form_ProgressBlue.XAML`).

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Variables should be pipe separated (|) and are referenced in the Theme as %1% %2% etc.

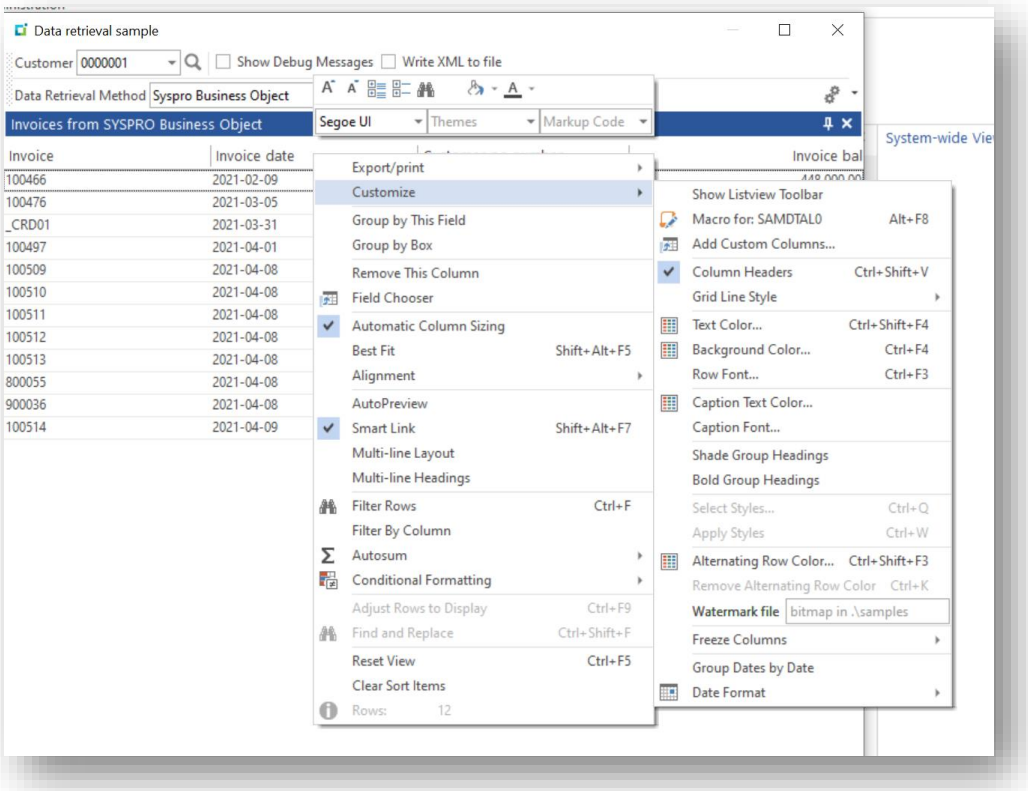
Some of these XAML themes have equivalent HTML themes so that they work in both SYSPRO and in Avanti – these are shown in the drop-down list when the **Show all themes** option is disabled.

An example of a list view using themes:

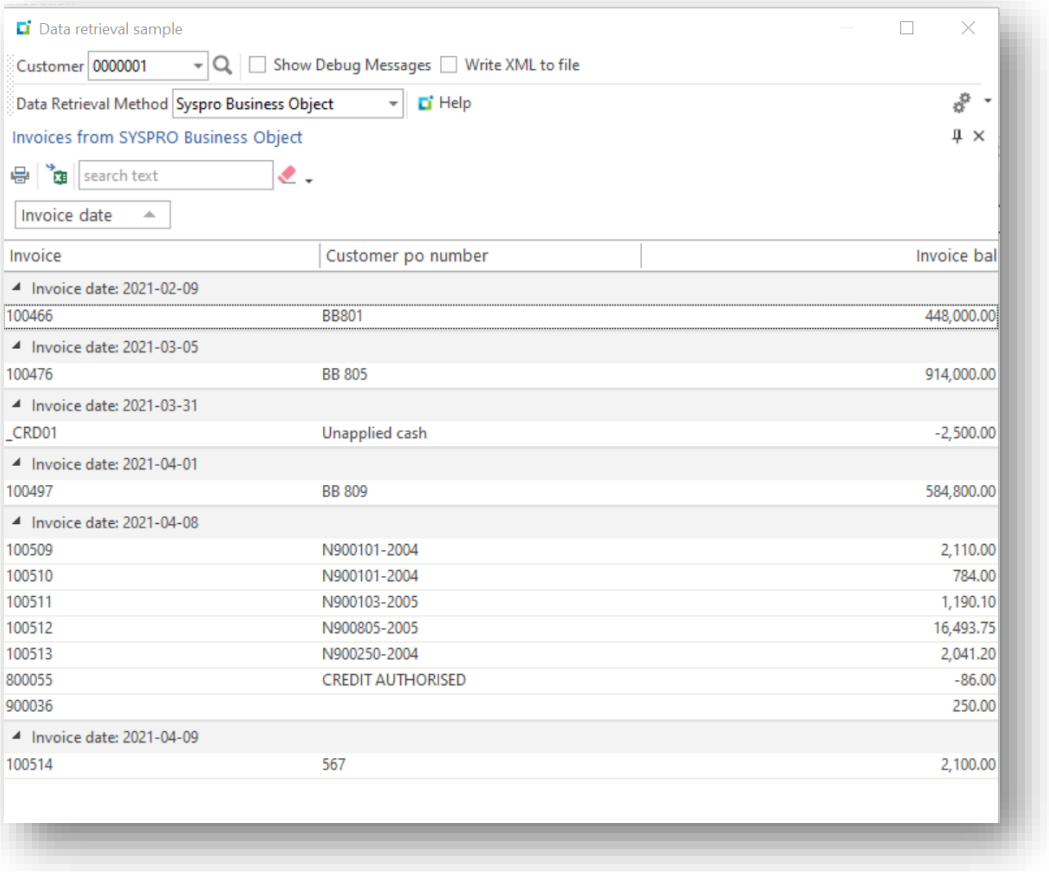


LIST VIEW BEHAVIOR

The list view in the Application Designer is the same control that the standard SYSPRO system uses and has all the same behaviors such as sorting, filtering, and formatting.



© SYSPRO. All Rights Reserved. All trademarks are recognized.



RETRIEVING LIST VIEW CONTENT

There are two ways of returning the contents of a list view from a script using the callouts **`ListviewGetRecords`** and **`ListviewGetRecordsAsXML`**:

LISTVIEWGETRECORDS

This callout returns a 2-dimensional array of rows and columns.

Here's example code that shows how to process this array:

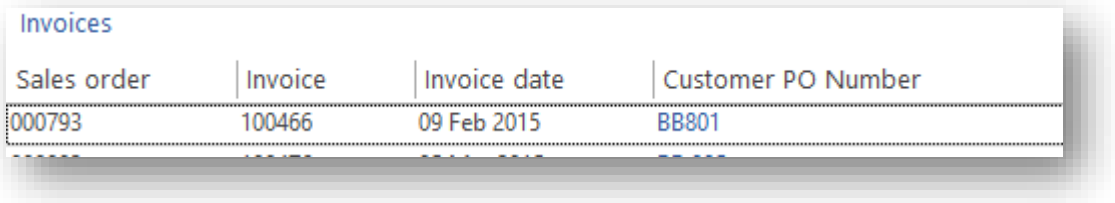
```
dim ReturnValue, rowindex, columnIndex
ReturnValue = CallSYSPROFunction(ListviewGetRecords, "DTS004L0", " ")
For rowindex = 0 to ubound(ReturnValue) -1
  For columnindex = 0 to ubound(ReturnValue(rowindex))
    Msgbox returnValue(rowindex)(columnindex)
  Next
Next
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

ListViewGetRecordsAsXML

This callout returns an XML string.

If the list view contents look like this:



then this will be the XML output:

```
<Records><Record><SalesOrder>000793</SalesOrder><Invoice>100466</Invoice>
<InvoiceDate>2015/02/09</InvoiceDate>
<CustomerPONumber>BB801</CustomerPONumber></Record></Records>
```

Form Control

The Form control used by the Designer is the primary User Interface control that the main SYSPRO application uses and as such inherits all the SYSPRO user interface look, feel and behavior. This means that any application using this control will look the same as the core SYSPRO product.

Adding a Form control to the application creates a Form Pane. There can be multiple forms in an application and each form can have its own toolbar.

Refer to the toolbar section for more detail.

Once a form pane has been created, the form content must be designed. The layout for all the form panes is the same. It is a two-column layout with a form field on each row. Column one for the description/caption and column two for the data.

The **Design Form** function enables the addition of an extensive list of field types to create the form.

Adding the Form Control

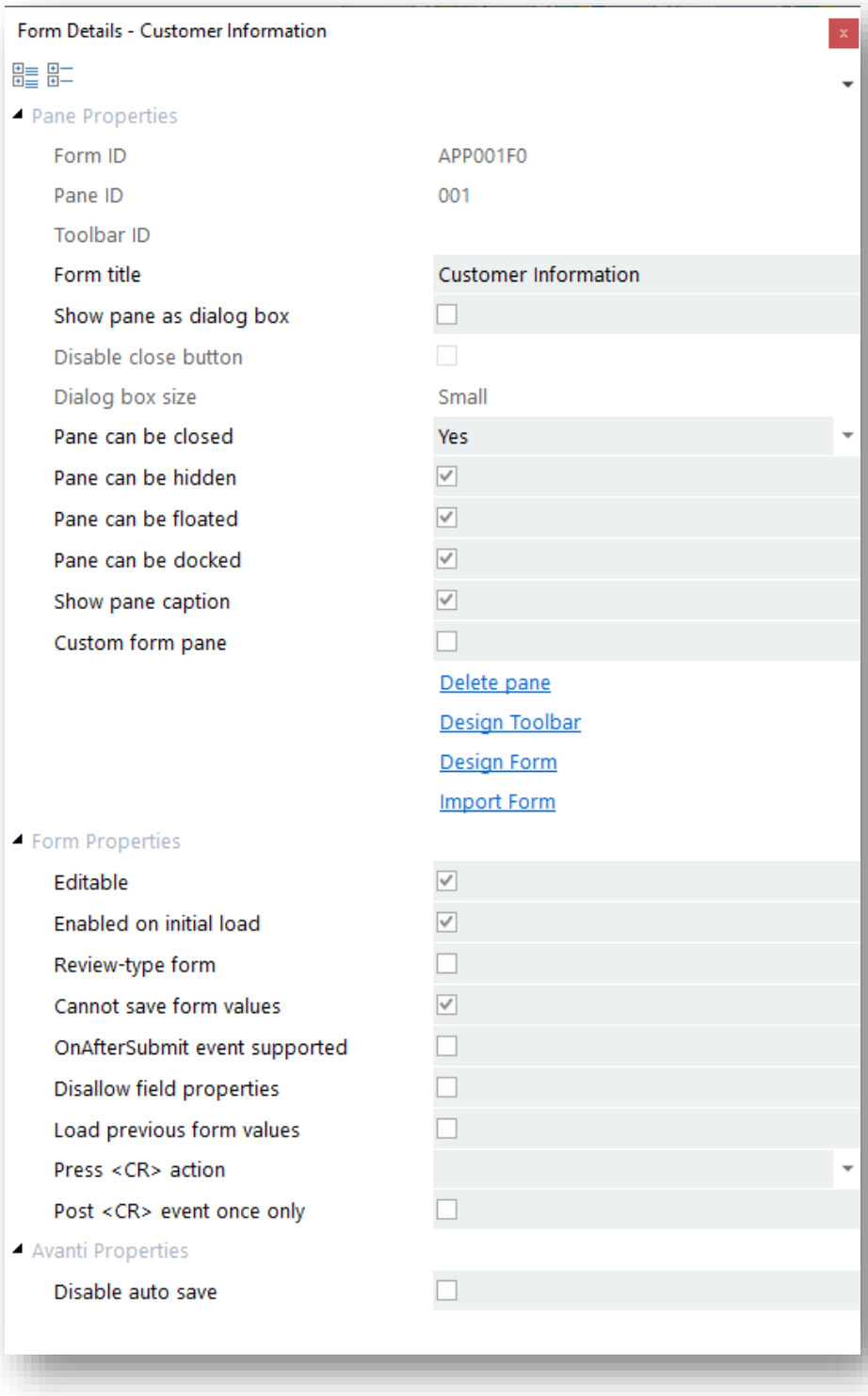
Clicking on the Form Control in the Toolbox will add a form pane.

The first decision is to decide if the form is a dialog box. A dialog box is a Form pane that (when opened by VBScript code) must be closed by the user to continue with the application. It is often used to force data entry and to control the user flow through an application.

Next, you should decide if the form is to be editable, that is, is the form being used to display data only (e.g. an enquiry screen) or is it being used to enter and amend data?

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Note that the form pane must be **enabled** to become editable. This can be done by VBScript or by the Form property **Enabled on initial load**:

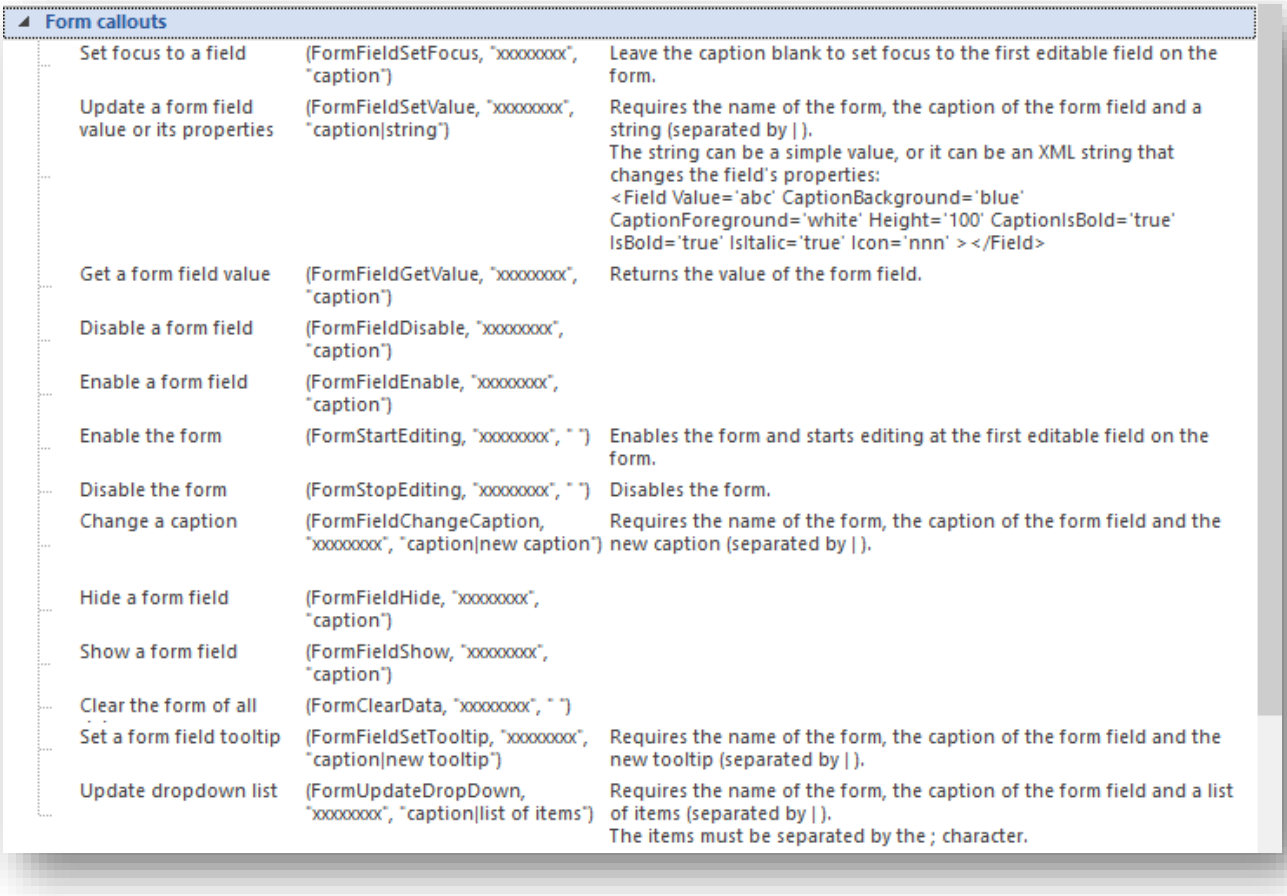


© SYSPRO. All Rights Reserved. All trademarks are recognized.

VBSCRIPT CALLOUTS

There is an extensive list of VBScript callouts to control the form and its fields.

Any field value can be read into the VBScript or set by the VBScript using these callouts:



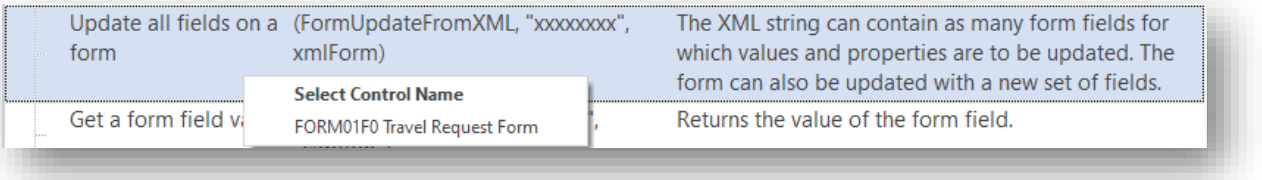
Form callouts		
Set focus to a field	(FormFieldSetFocus, "xxxxxxxx", "caption")	Leave the caption blank to set focus to the first editable field on the form.
Update a form field value or its properties	(FormFieldSetValue, "xxxxxxxx", "caption|string")	Requires the name of the form, the caption of the form field and a string (separated by |). The string can be a simple value, or it can be an XML string that changes the field's properties: <Field Value='abc' CaptionBackground='blue' CaptionForeground='white' Height='100' CaptionIsBold='true' IsBold='true' IsItalic='true' Icon='nnn' ></Field>
Get a form field value	(FormFieldGetValue, "xxxxxxxx", "caption")	Returns the value of the form field.
Disable a form field	(FormFieldDisable, "xxxxxxxx", "caption")	
Enable a form field	(FormFieldEnable, "xxxxxxxx", "caption")	
Enable the form	(FormStartEditing, "xxxxxxxx", " ")	Enables the form and starts editing at the first editable field on the form.
Disable the form	(FormStopEditing, "xxxxxxxx", " ")	Disables the form.
Change a caption	(FormFieldChangeCaption, "xxxxxxxx", "caption|new caption")	Requires the name of the form, the caption of the form field and the new caption (separated by |).
Hide a form field	(FormFieldHide, "xxxxxxxx", "caption")	
Show a form field	(FormFieldShow, "xxxxxxxx", "caption")	
Clear the form of all	(FormClearData, "xxxxxxxx", " ")	
Set a form field tooltip	(FormFieldSetTooltip, "xxxxxxxx", "caption|new tooltip")	Requires the name of the form, the caption of the form field and the new tooltip (separated by |).
Update dropdown list	(FormUpdateDropDown, "xxxxxxxx", "caption|list of items")	Requires the name of the form, the caption of the form field and a list of items (separated by |). The items must be separated by the ; character.

Most of the callouts are self-explanatory but the callout **FormUpdateFromXML** may require further explaining.

FORMUPDATEFROMXML CALLOUT

This callout can populate a form with values derived from an XML string.

This allows you to populate an entire form with a single callout instead of having to use the **FormFieldSetValue** callout for multiple fields.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

When the Form control name is selected, the script editor will construct the XML string consisting of the names of all the fields on the form and insert it into the script.

You just have to define the values for each form field:

```
dim xmlForm
xmlForm="<Form ClearData='true' ClearForm='false'><Fields>" &_
"<Field Name='Date requested' Value=' ' />" &_
"<Field Name='First name' Value=' ' />" &_
"<Field Name='Last name' Value=' ' />" &_
"<Field Name='Date of birth' Value=' ' />" &_
"<Field Name='Destination' Value=' ' />" &_
"<Field Name='Departure date' Value=' ' />" &_
"<Field Name='Travel purpose' Value=' ' />" &_
"<Field Name='Currency' Value=' ' />" &_
"<Field Name='Airfare' Value=' ' />" &_
"<Field Name='Transportation' Value=' ' />" &_
"<Field Name='Hotel accommodation' Value=' ' />" &_
"</Fields></Form>"
returnValue = CallSYSPROFunction(FormUpdateFromXML, "FORM01F0", xmlForm)
```

The XML element **ClearData** indicates if the form is to be cleared of all data before any fields are updated with values. The default is 'true', but if set as 'false' then the form is not cleared of any data first.

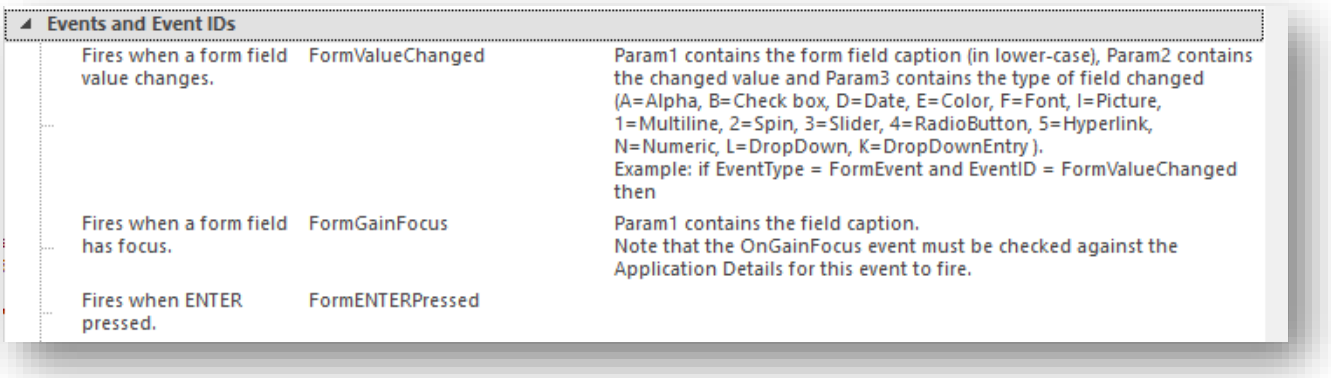
The XML element **ClearForm** indicates if the form is to be completely cleared of all form fields. Using a value of 'true' allows you to dynamically build a form on-the-fly.

returnValue will be blank if the incoming XML is well-formed.

If the XML is not well-formed then **returnValue** will contain an error message.

Form Events

The form raises the following events:



Events and Event IDs		
Fires when a form field value changes.	FormValueChanged	Param1 contains the form field caption (in lower-case), Param2 contains the changed value and Param3 contains the type of field changed (A=Alpha, B=Check box, D=Date, E=Color, F=Font, I=Picture, 1=Multiline, 2=Spin, 3=Slider, 4=RadioButton, 5=Hyperlink, N=Numeric, L=DropDown, K=DropDownEntry). Example: if EventType = FormEvent and EventID = FormValueChanged then
Fires when a form field has focus.	FormGainFocus	Param1 contains the field caption. Note that the OnGainFocus event must be checked against the Application Details for this event to fire.
Fires when ENTER pressed.	FormENTERPressed	

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Field Types

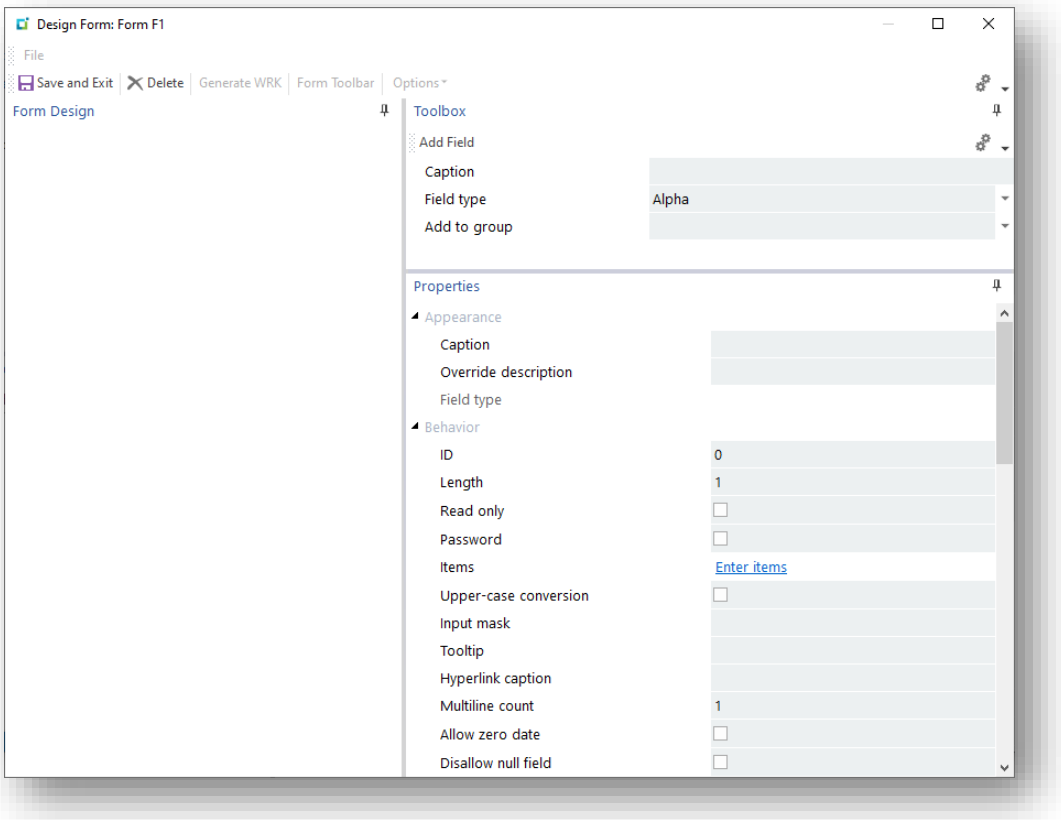
When the **FormValueChanged** event is raised, **Param3** holds the field type.

- A = Alpha
- B = Check box
- E = Color Picker
- D = Date
- L = Drop down
- L = Drop down Entry
- F = Font picker
- 5 = Hyperlink
- 1 = Multiline
- N = Numeric
- 4 = Radio Button
- 3 = Slider
- 2 = Spin Button
- 6 = Time
- 7 = Time span

The **Picture viewing** and **Group Heading collapse and expand** actions do not raise events.

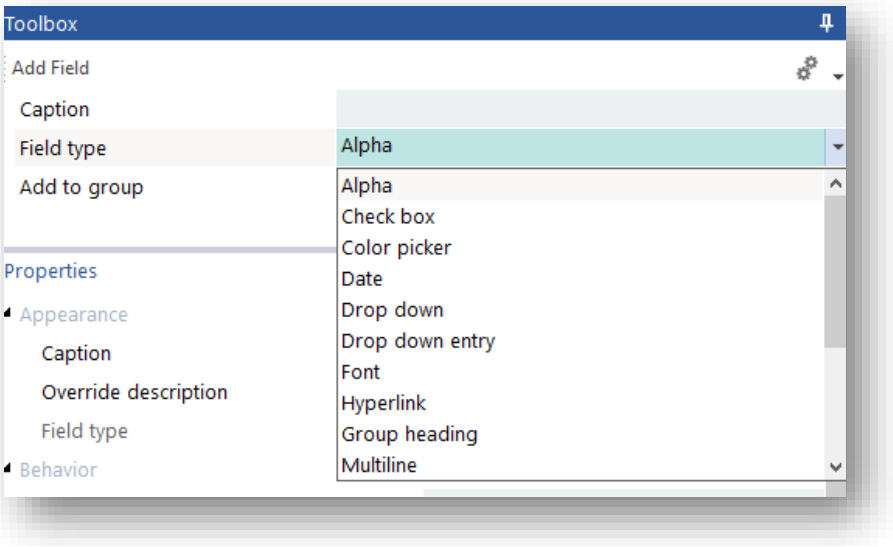
Form Design

Once the initial Form pane has been created the next step is to create the form content. Clicking on **Design Form** will launch the form design screen.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

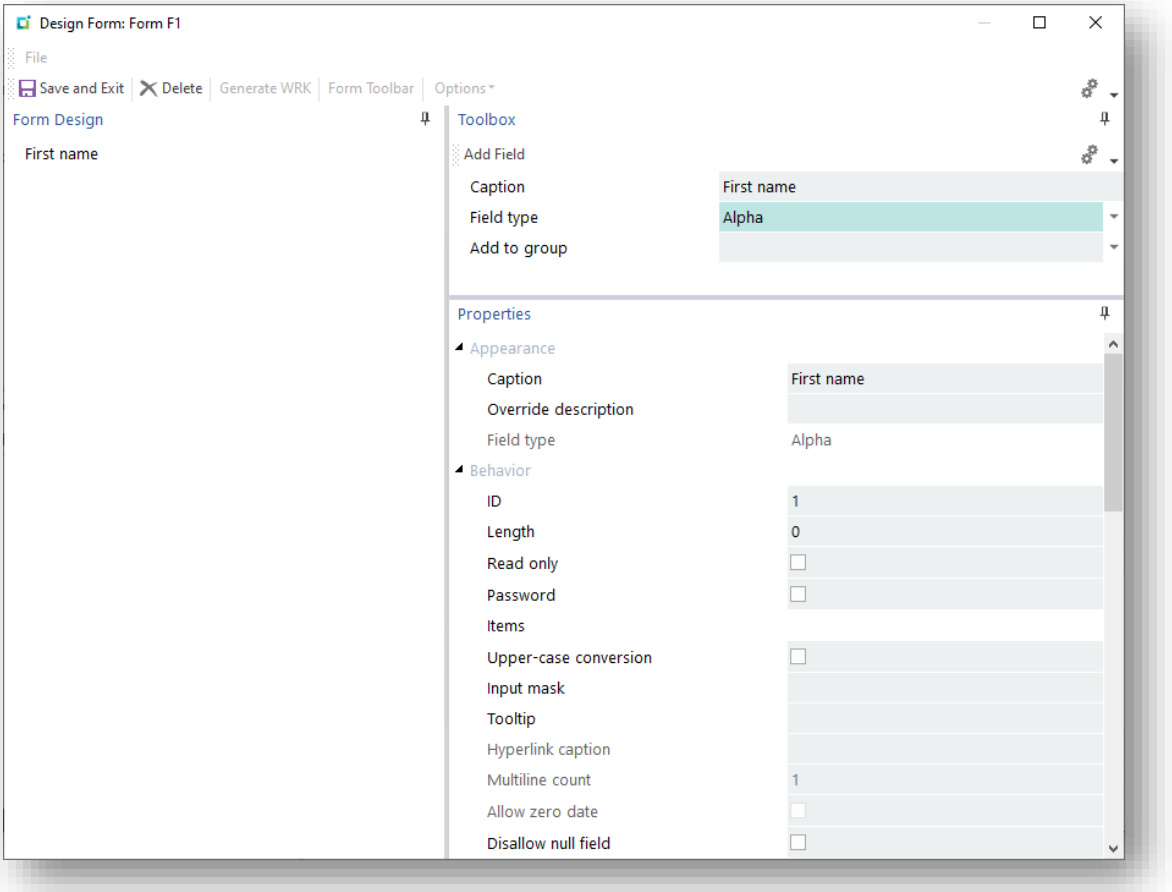
There is an extensive list of Field Types that can be added to a form and each Field Type has its own set of properties that can be set.



These are designed to alter its look and behavior.

Adding Fields

Enter a Caption, select a field type, decide if it needs to be added to an existing Group Header Control and then press the **Add Field** button on the toolbar. The field will appear in the Form Design window, it will be assigned a unique ID and at the same time the field properties window will be populated with the field details and any field specific properties enabled.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Caption Considerations

Please note that under normal circumstances, the Caption should be unique to a form.

If in the event it is necessary to have two fields on the form with the same caption, then you need to use the **Override Description** property to create a unique reference. The override description is then used in any VBScript. It should also be used if the caption is to be left blank.

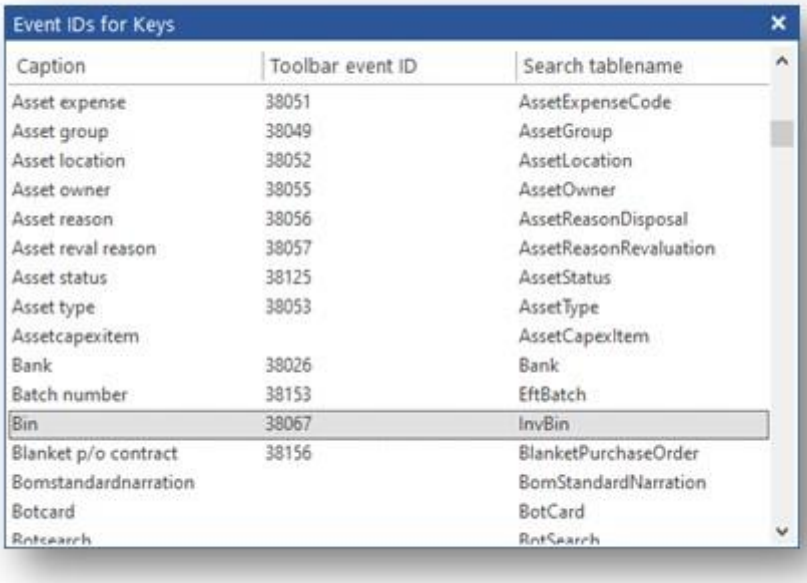
See the Online Help for more information.

If the caption matches the caption of a SYSPRO key field, then the field inherits the behavior of that key field. For example, setting an Alpha field caption as **Customer** will automatically add a browse button for a customer browse (and Predictive Search will be enabled) and when the application runs a right-click will display all the options available in SYSPRO when right-clicking a customer.

You can click on the **Event IDs for Keys** hyperlink in the Application Details to view all the available Captions for **key field**. This list will indicate what captions you need to use.

For example:

For a BIN LOCATION you'll need the word **Bin**.



Pressing **Save and Exit** on the Form design window returns you to the Application Designer main layout and displays the layout in the form pane.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Field Defaults

To set a field default, enter a value into the field whilst in design mode.

Alternatively, use VBScript and the `AppDesigner_OnInitialization()` function to set values on the form as it starts up.

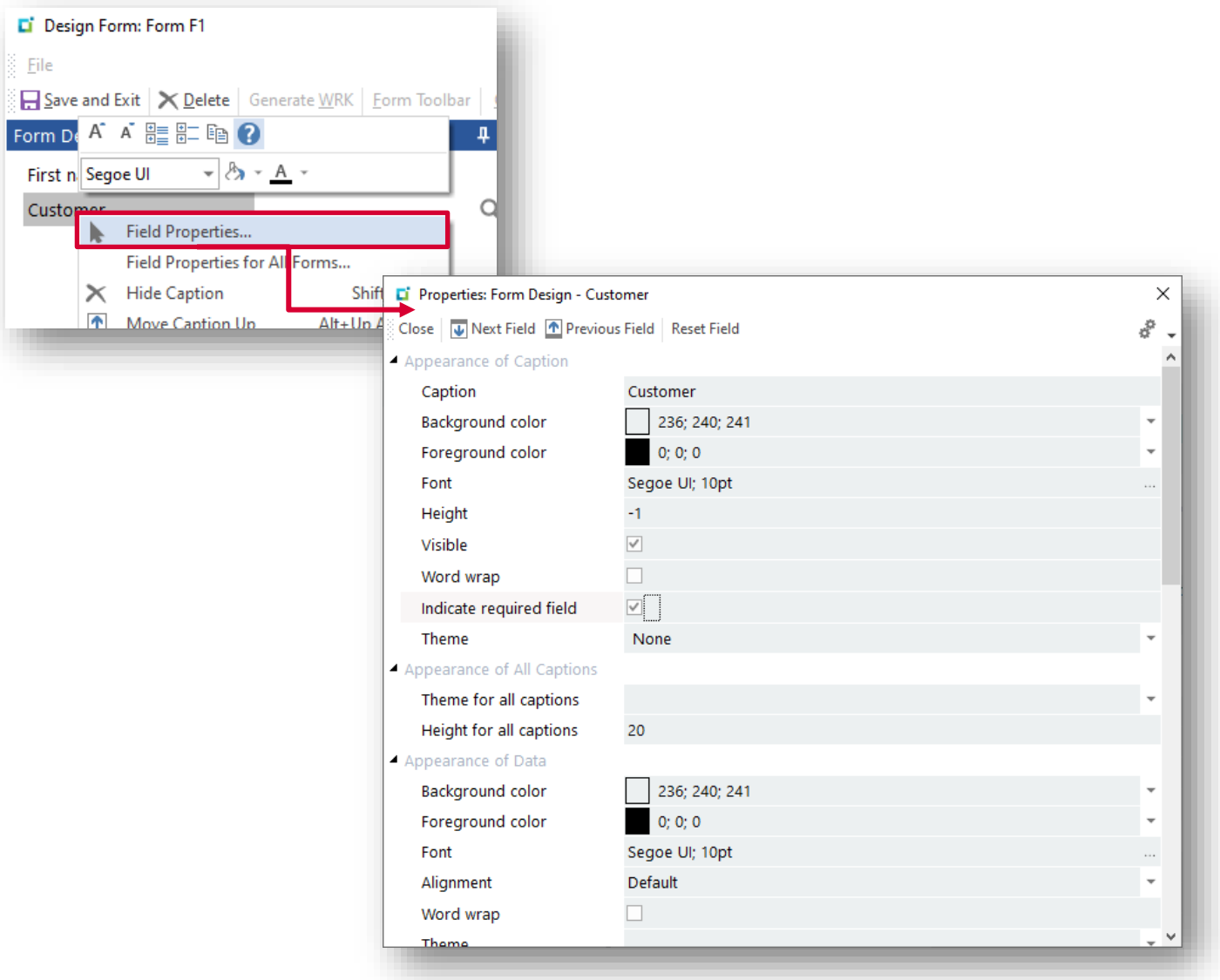
Re-ordering Fields

The order of the fields on the form can be changed by simply clicking on the required field with the mouse and dragging it up or down the form.

Alternatively, you can press `Alt+UpArrow` or `Alt+DownArrow` to move items up or down. This key also moves items into a Group category.

Setting Field fonts and colors

For more sophisticated formatting, right-click on any form field and select **Field Properties**:



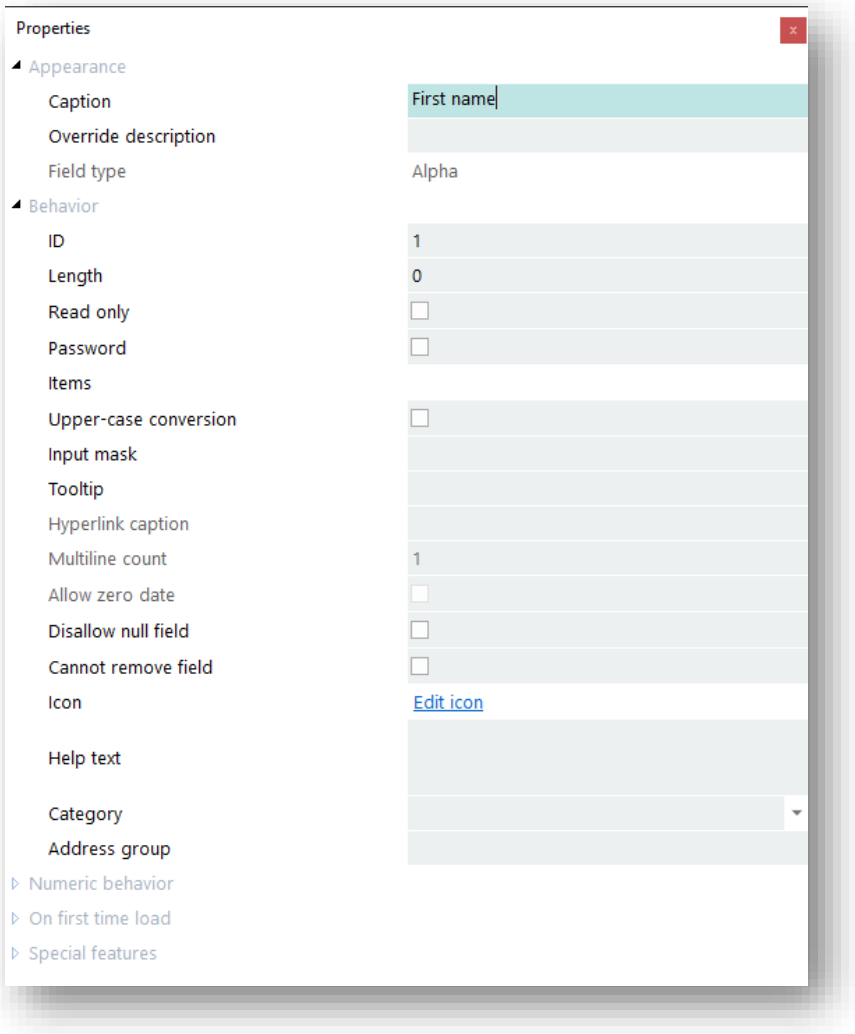
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Field Types

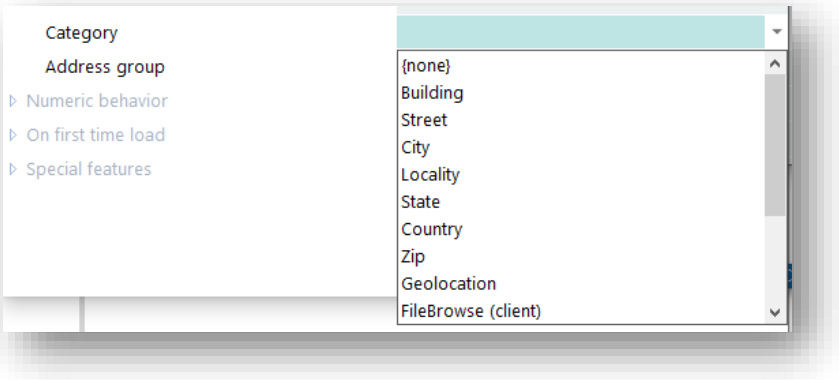
This section describes each of the field types that can be added to a Form Pane.

Alpha

The Alpha field type is the same as a traditional text box. Available properties include:



The Category property qualifies the type of Alpha field.



For example, setting the category to **FileBrowse (client)** will display a browse button next to the field which (when pressed) will display a file browser browsing the local machine.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Checkbox

The checkbox is either checked or not. A value of `1` is checked and `0` is unchecked.

There is a Special features property that allows this to change to Y and N. This is particularly useful when using SYSPRO business objects to populate a form because they use Y and N for checkbox type questions.

If you need to set a default value for the checkbox, then simply check it or uncheck it in the form design.

Color Picker

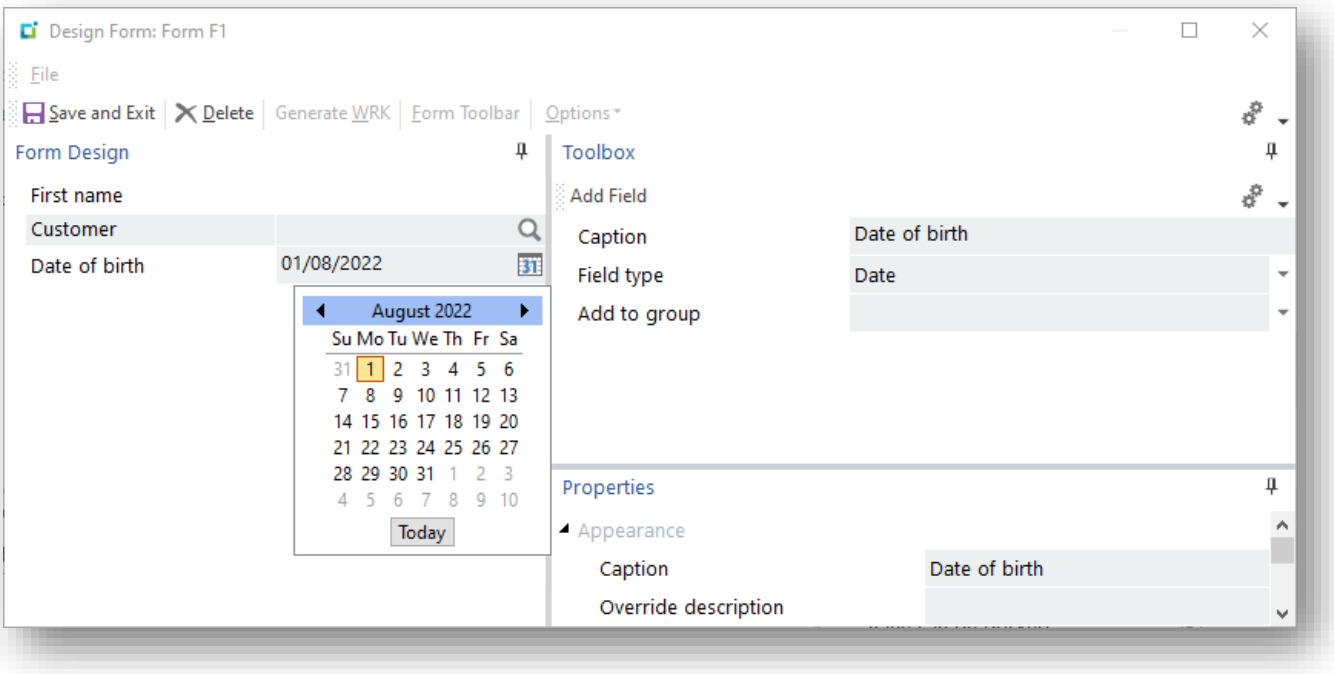
This control allows the selection of an RGB color sequence.

A useful color conversion site is: https://www.rapidtables.com/convert/color/rgb-to-hex.html

Date

A date control that allows the selection of a date from a date picker.

If setting the value of this field via VBScript, then the format must be `CCYYMMDD` (e.g. `20220620`).



The tooltip for this control holds a full description of the date.

See VBScript hints for creating a date in this format.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

DROP DOWN

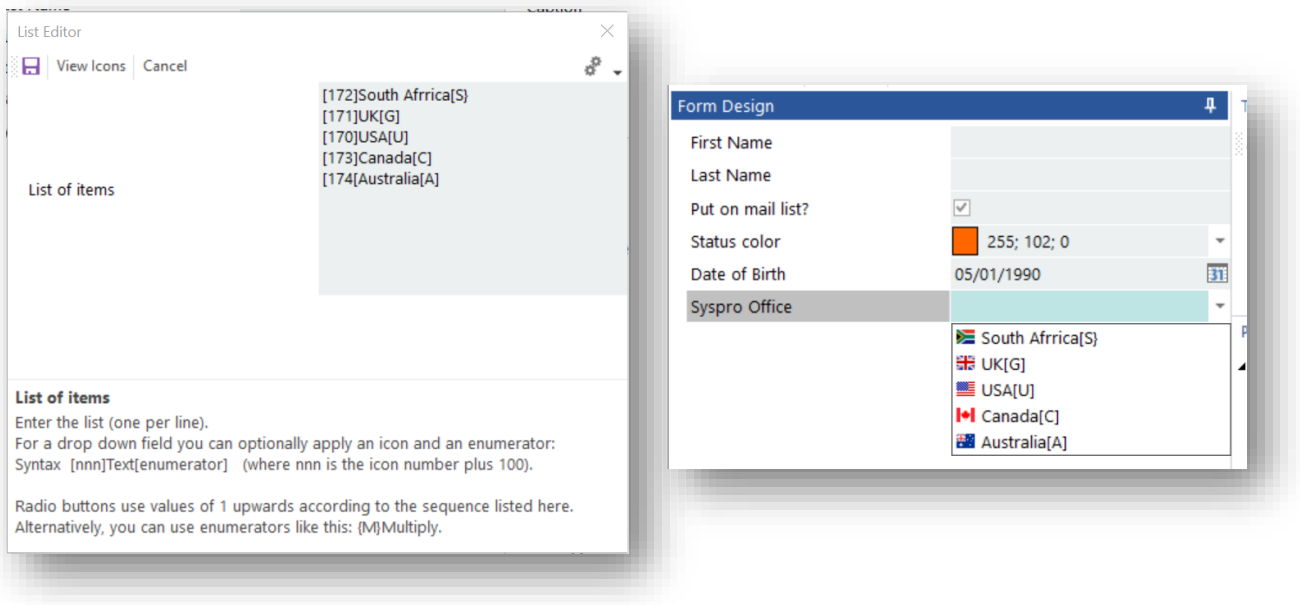
The drop-down field type enables a fixed list of options to be presented.

To enter items in the list, select the **Enter Items** link in the **Behavior** section of the properties. A dialog pane will appear enabling the entry of items in the list.

The list can have an icon, text and an enumerator specified.

- The icon and text will be displayed in the drop down.
- The enumerator can be used to override the index of the drop down.
  If used, then it will return the enumerator to the VBScript rather than the index number of the item in the list.

In design mode, the enumerator is displayed. However, when the form runs in the application, the enumerator is not displayed.



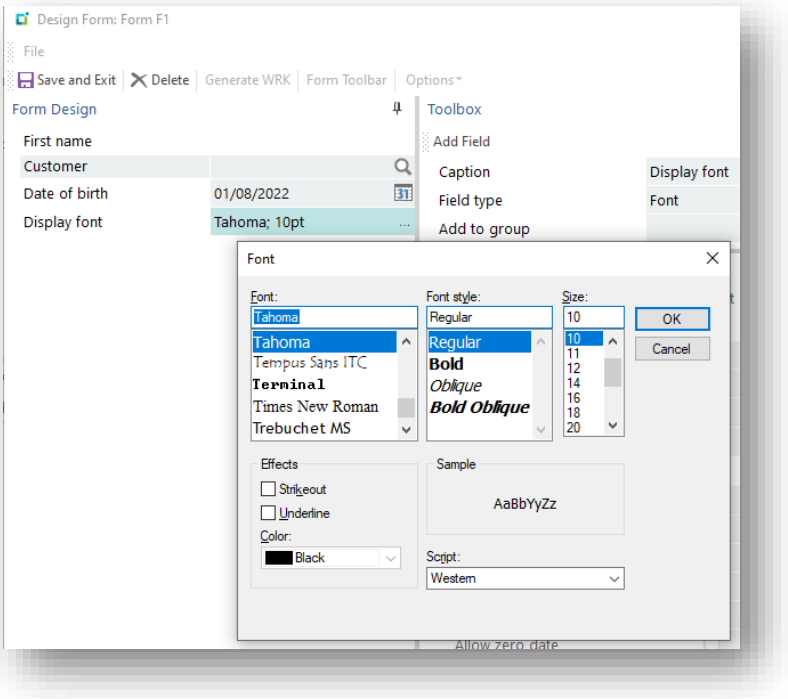
DROP DOWN ENTRY

The drop-down entry field behaves the same as the drop-down field type, except that it allows the user to enter data that is not in the drop-down list.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

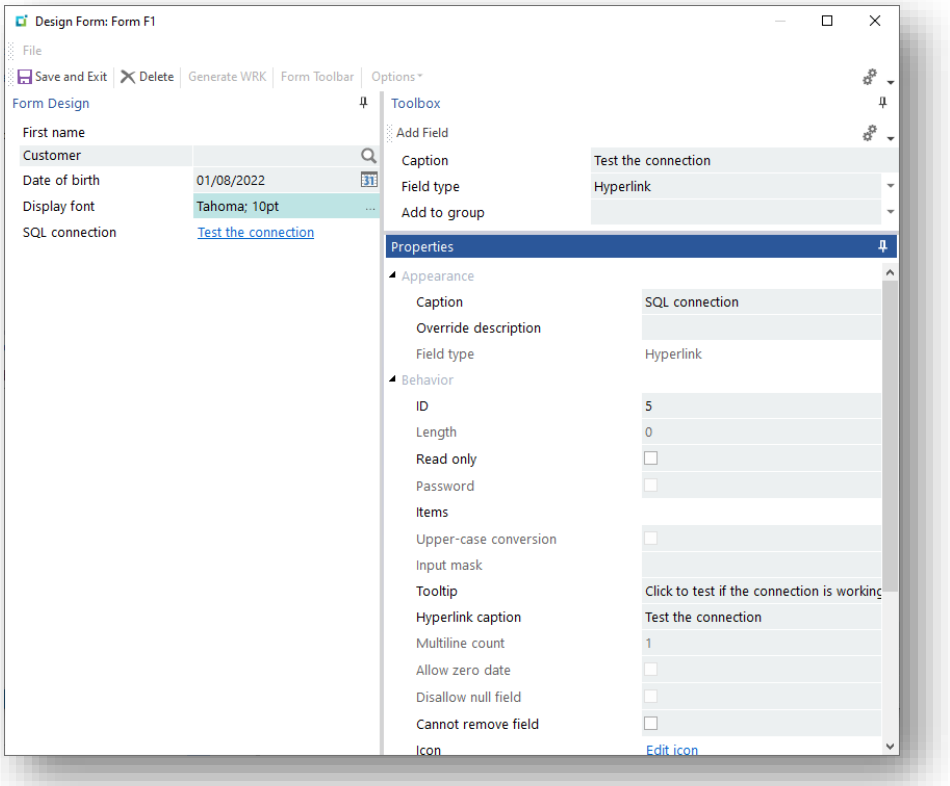
FONT

The Font field type shows a Font picker field. This could be used in VBScript to change the look and feel of different parts of the form:



HYPERLINK

Displays the caption as a hyperlink.



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Group Heading

This field allows the grouping of fields into a collapsible section.

Once created, it will appear in the **Add to Group** drop down field when creating other fields.

Fields can be moved into a group heading when in design mode by dragging a field to the group. You can also press `Alt+UpArrow` to nudge a field into a group that is above the field, or `Alt+DownArrow` to move into a group that is below.

The group can be collapsed automatically by setting the **Show Group Collapsed** property.

Multiline

The multiline field allows the display and entry of a multiline text field. The number of lines can be set in the **Multiline count** property.

Numeric

This field type is a numeric field. The properties enable the setting of the overall length, the number of decimals and whether to allow commas when editing.

Picture

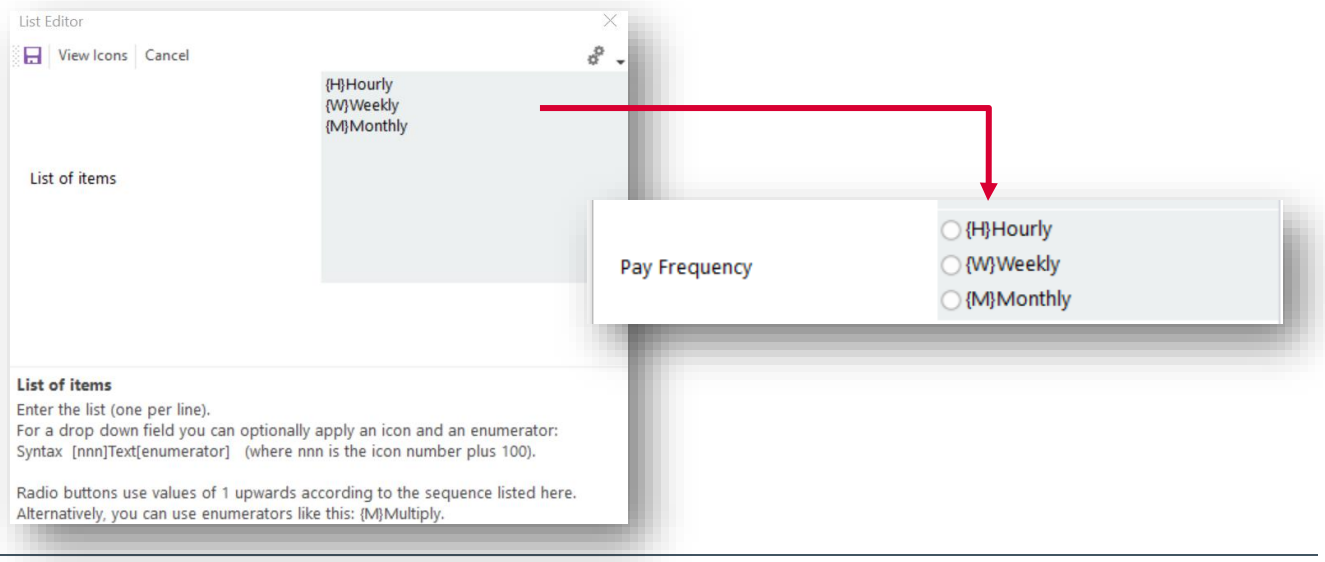
This field type displays a picture. The text in the field should be a URL to a picture. When the form runs, the picture is displayed and (if selected) will be shown in the default picture viewer.

Radio Button

This field type displays a list of items as a series of radio buttons allowing the selection of a single option. The radio button list is created by selecting the **Enter items** option.

By default, the value returned to the VBScript is an index number of the item in the list, but an enumerator can be set to override this and return an alpha value to the script.

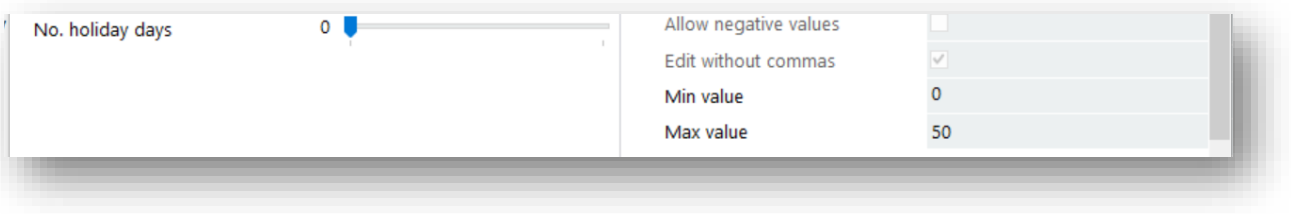
The enumerator is displayed in the designer but not when the application runs:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Slider

This field type is a numeric field that can be set by sliding a pointer along a slider. Maximum and Minimum values can be set.



Spin Button

This field type is a numeric field that can be increased and decreased by arrows of a spin control. Maximum and Minimum values can be set.

Time

A 24-hour time format field allowing entry and display of `hh:mm` data.

Hours and minutes are validated automatically.

Time Span

Display days, hours, minutes format.

VBScript Hints

Setting Field Values

When setting field values, the caption is separated from the value by a pipe (“|”) sign.

If this is omitted, then the field will not be populated.

For example:

```
call CallSYSPROFunction(FormFieldSetValue, "FRMTSTF0", "Name|" & strName)
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

SETTING A DATE VALUE

A Date field must be in the **CCYYMMDD** format.

- Both the Month and Day must be two characters
- Single digit numbers must have a leading zero

An example of building a date in this format is:

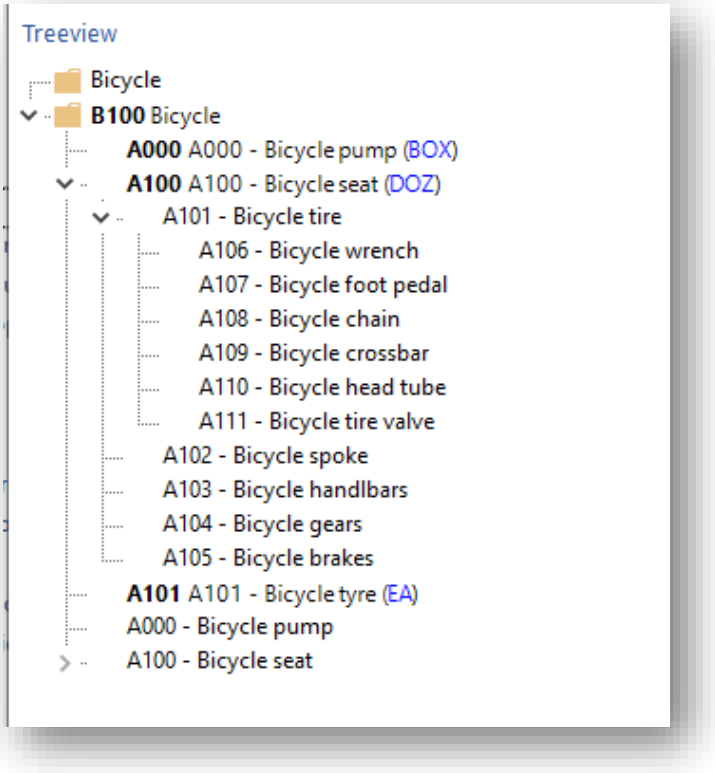
```
dim ccyy: ccyy = year(date)
dim mm : mm = right(string(2,"0") & month(date),2)
dim dd : dd = right(string(2,"0") & day(date),2)
dim ccyymmdd: ccyymmdd = ccyy & mm & dd
```

```
call CallSYSPROFunction(FormFieldSetValue, "ANDIEXF2", "Transaction date|" & ccyymmdd)
```

TREE VIEW CONTROL

The Tree View control is used to display hierarchical representations of items, similar to the way that the files and folders are displayed in the left pane of the Windows Explorer.

Each node may contain one or more child nodes:



These notes explain some of the basics behind the Tree view control.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

How it works

The following explains the basic process of a tree view creation:

1. A tree view is built up by adding “Items” to the tree control.
2. Each item has a unique “node index” that the Application Designer automatically allocates to the item when its added.
3. The root node/item of a tree view is always node index **1** and has a Parent attribute of **0** (i.e. it doesn’t have a parent).
4. All other “items” must belong to a parent.

For example:

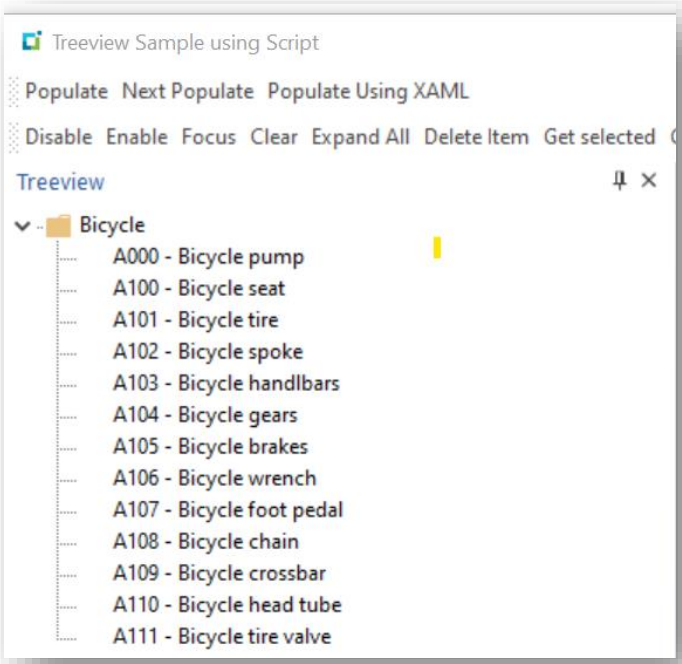
The following function adds the first level of a tree view:

```
function PopulateTreeview()

dim xmlString
xmlString = "<Items>" & _
"<Item Parent='0' Caption='Bicycle' Key='B100' HasChildren='Y' Icon='5' />" &_
"<Item Parent='1' Caption='A000 - Bicycle pump' Value='A000' />" &_
"<Item Parent='1' Caption='A100 - Bicycle seat' Value='A100' />" &_
"<Item Parent='1' Caption='A101 - Bicycle tire' Value='A101' />" &_
"<Item Parent='1' Caption='A102 - Bicycle spoke' Value='A102' />" &_
"<Item Parent='1' Caption='A103 - Bicycle handlbars' Value='A103' />" &_
"<Item Parent='1' Caption='A104 - Bicycle gears' Value='A104' />" &_
"<Item Parent='1' Caption='A105 - Bicycle brakes' Value='A105' />" &_
"<Item Parent='1' Caption='A106 - Bicycle wrench' Value='A106' />" &_
"<Item Parent='1' Caption='A107 - Bicycle foot pedal' Value='A107' />" &_
"<Item Parent='1' Caption='A108 - Bicycle chain' Value='A108' />" &_
"<Item Parent='1' Caption='A109 - Bicycle crossbar' Value='A108' />" &_
"<Item Parent='1' Caption='A110 - Bicycle head tube' Value='A109' />" &_
"<Item Parent='1' Caption='A111 - Bicycle tire valve' Value='A110' />" &_
"</Items>"
call CallSYSPROFunction(TreeAddItems, "TREEVWX0", xmlString)
end function
```

This code creates a root item of **Bicycle**. It has a node index of **1** and no parent.

All the other items are its children. Their **Parent** attribute is set to **1** (which is the node index of the root node):

© SYSPRO. All Rights Reserved. All trademarks are recognized.



This can be represented like this:

Caption	Parent ID	Node Index	Children
Bicycle	0	1	1
A000 - Bicycle pump	1	2	0
A100 - Bicycle seat	1	3	0
A101 - Bicycle tire	1	4	0
A102 - Bicycle spoke	1	5	0
A103 - Bicycle handlebars	1	6	0
A104 - Bicycle gears	1	7	0
A105 - Bicycle brakes	1	8	0
A106 - Bicycle wrench	1	9	0
A107 - Bicycle foot pedal	1	10	0
A108 - Bicycle chain	1	11	0
A109 - Bicycle crossbar	1	12	0
A110 - Bicycle head tube	1	13	0
A111 - Bicycle tire valve	1	14	0

© SYSPRO. All Rights Reserved. All trademarks are recognized.

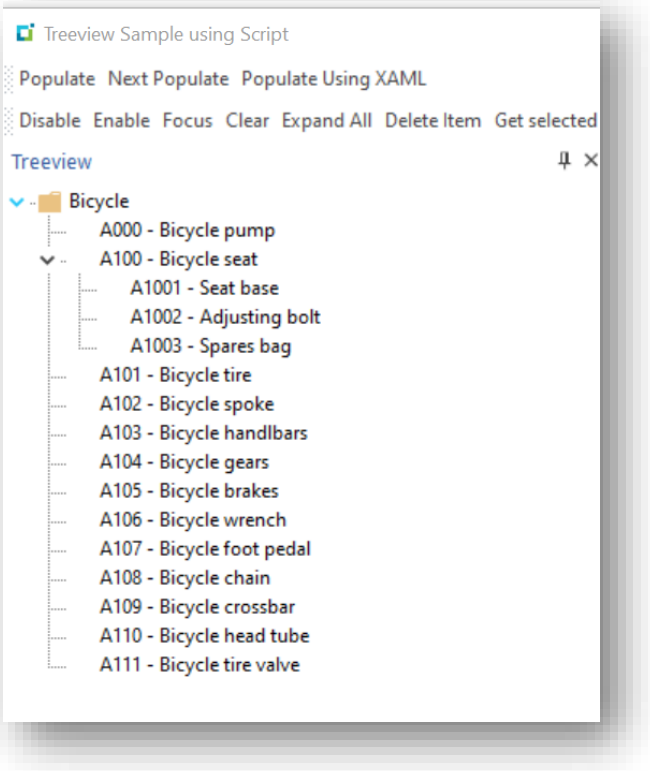
The following indicates how to add children to a node like this:

```
function PopulateTreeview2()

dim xmlString
xmlString = "<Items>" & _
"<Item Parent='3' Caption='A1001 - Seat base' Value='A1001' />" &_
"<Item Parent='3' Caption='A1002 - Adjusting bolt' Value='A1002' />" &_
"<Item Parent='3' Caption='A1003 - Spares bag' Value='A1003' />" &_
"</Items>"
call CallSYSPROFunction(TreeAddItems, "TREEVWX0", xmlString)
end function
```

This code adds 3 child items to node index **3**, which is the bicycle seat.

As it does so, it sets the **HasChildren** attribute of the new parent to **Y**.

Each new item is allocated a unique node index:

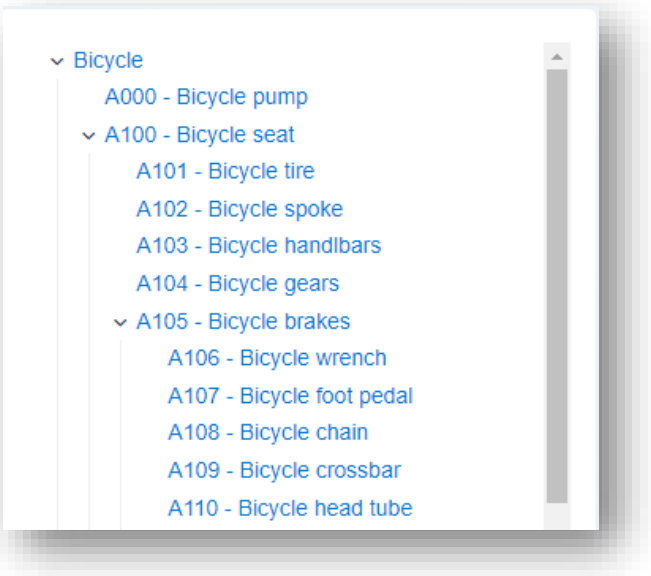


This can be represented like this:

Caption	Parent ID	Node Index	Children
A100 - Bicycle seat	1	3	1
A1001 - Seat base	3	15	0
A1002 - Adjusting bolt	3	16	0
A1003 - Spares bag	3	17	0

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Note: If you wish the tree view to appear the same in Avanti, then you should check the tree view option **Multi-level tree view**:



Application Designer Tree view Events

Every Tree view event returns the following information to the VBScript **OnUserEvent**:

```
Function AppDesigner_OnUserEvent(EventType,EventName,EventID,Param1,Param2,Param3)
```

*See **Event Types** and **Events & Event Ids** on the **Callout Functions** tab in VBScript.*

These parameters are explained as follows:

Parameter	Explanation
EventType	This is the "TreeviewEvent" constant, identifying what category of event it is.
EventName	This is the TreeviewID. This identifies which Tree view the events are for. See its Pane Properties
EventID	This will be the event that has just fired (for example: TreeExpand, TreeClick, TreeRightClick, etc.).
Param1	This returns whatever string is in the "Value" attribute
Param2	This returns the node index of the item that invoked the event. E.g. the item that was clicked. This would be used as the parent attribute if adding children.
Param3	This returns: ▪ 1 = Has Children ▪ 0 = Has no children

Note: The Value attribute can hold any string. If you need to return more than one value from the tree view, then build it into the value attribute when you build the tree view.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

An example taken from the SAMCOC sample app:

```
function PopulateTreeview( m_fullurl,m_alphabet, m_listby)
'NB The value attribute on an <Item> is always returned to the VBScript on param1
of any Treeview event.
'In this example I have two fields delimited by a '|' sign. The word "Cat" is
used to identify an item that is a category.
'I use this when deciding on whether to show a context menu or not. I only want a
context menu on a drink and not a category.
….
….
        for each drink in oDrinks
            TreeviewXML = TreeviewXML & "<Item Parent='1' Caption='" &
drink.strCategory  & "' Value='" & "Cat|" & drink.strCategory &  "'
HasChildren='Y'/>"
        next
```

OTHER POINTS

To all intents and purposes there is no limit to the size of a tree view.

You can build a complete tree view with levels of parents and children for an initial load. However, data volumes for an application may make this not feasible.

Realistically, it is best practice to get the branches of a tree as and when you need them. Typically, this occurs when you click a node to expand it.

XAML AND TREE VIEW ITEMS

XAML can also be injected into a Tree view item.

To help simplify passing a XAML string to a tree view item and having to substitute the XML special characters, if the **<** and **>** characters are replaced with **{** and **}**, then the Application Designer will do the substitution for you.

Note: It will not substitute any other special characters that may be in the text, such as Ampersand (&).

If provided as a string to an attribute for a control, you need to make sure that any XML reserved characters are replaced with their relevant encoding. Otherwise XAML and XML get mixed up and nothing will work.

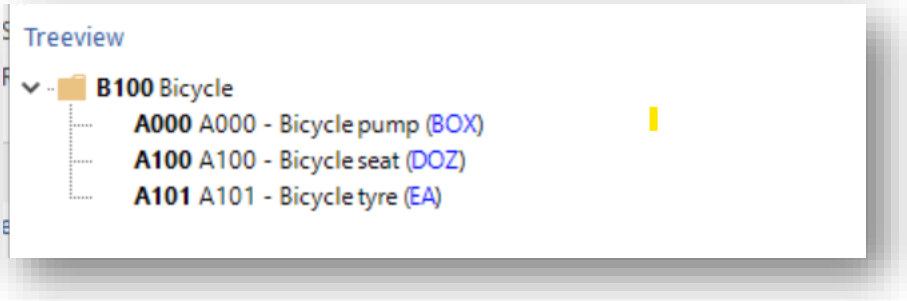
There is a code snippet provided in the Application Designer called **`CheckXMLSpecialCharacters`** which will parse a string and substitute the relevant codes if required.

Therefore, the following code:

© SYSPRO. All Rights Reserved. All trademarks are recognized.

```
dim xmlString
xmlString = "<Items>" _
          & "<Item  XAML='&lt;TextBlock&gt;&lt;Bold&gt;B100&lt;/Bold&gt; Bicycle&lt;/TextBlock&gt;'  Parent='0' Caption='Bicycle' Value='B100'  HasChildren='Y' Icon='5' />" _
          & "<Item Parent='1' Caption='A000 - Bicycle pump' Value='A000' " _
          & "XAML='{TextBlock}{Bold}A000{/Bold} A000 - Bicycle pump ({Run Foreground=""Blue"" Text=""BOX""/}){/TextBlock}'/>" _
          & "<Item Parent='1' Caption='A100 - Bicycle seat' Value='A100' " _
          & "XAML='{TextBlock}{Bold}A100{/Bold} A100 - Bicycle seat ({Run Foreground=""Blue"" Text=""DOZ""/}){/TextBlock}'/>" _
          & "<Item Parent='1' Caption='A101 - Bicycle tyre' Value='A101' " _
          & "XAML='{TextBlock}{Bold}A101{/Bold} A101 - Bicycle tyre ({Run Foreground=""Blue"" Text=""EA""/}){/TextBlock}'/>" _
          & "</Items>"

call CallSYSPROFunction(TreeAddXAMLItems, "TREEVWX0", xmlString)
```

Generates this:



SYNTAX EDITOR CONTROL

The Syntax editor control provides users with a highly sophisticated text editor control that supports many advanced features, including (but not restricted to):

- Text block grouping
- Syntax colorization
- Line numbers
- Font type
- Pre-defined color schemes (SQL, VBScript, XAML, XML)
- Bookmarks
- Break points

The editor is the same one used for the VBScript editor. And using the control is straight forward.

The Callout functions below explain the functionality available.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

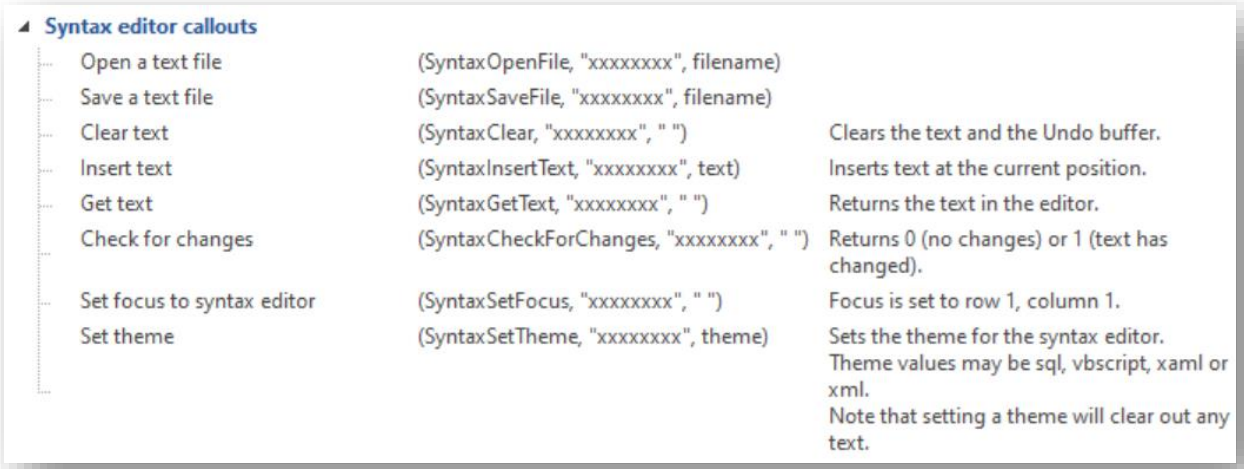
Features of the control

A mini toolbar (as part of the control) allows the user the following capabilities:

- Cut, Copy and Paste text
- Undo and Redo actions
- Find text
- Zoom text (and print)
- Print text
- Tidy an XML/XAML string
- Enlarge and reduce font size

Callout Functions

The Application Designer has the following callout functions for the Syntax Editor:



Syntax editor callouts		
Open a text file	(SyntaxOpenFile, "xxxxxxxx", filename)	
Save a text file	(SyntaxSaveFile, "xxxxxxxx", filename)	
Clear text	(SyntaxClear, "xxxxxxxx", " ")	Clears the text and the Undo buffer.
Insert text	(SyntaxInsertText, "xxxxxxxx", text)	Inserts text at the current position.
Get text	(SyntaxGetText, "xxxxxxxx", " ")	Returns the text in the editor.
Check for changes	(SyntaxCheckForChanges, "xxxxxxxx", " ")	Returns 0 (no changes) or 1 (text has changed).
Set focus to syntax editor	(SyntaxSetFocus, "xxxxxxxx", " ")	Focus is set to row 1, column 1.
Set theme	(SyntaxSetTheme, "xxxxxxxx", theme)	Sets the theme for the syntax editor. Theme values may be sql, vbscript, xaml or xml. Note that setting a theme will clear out any text.

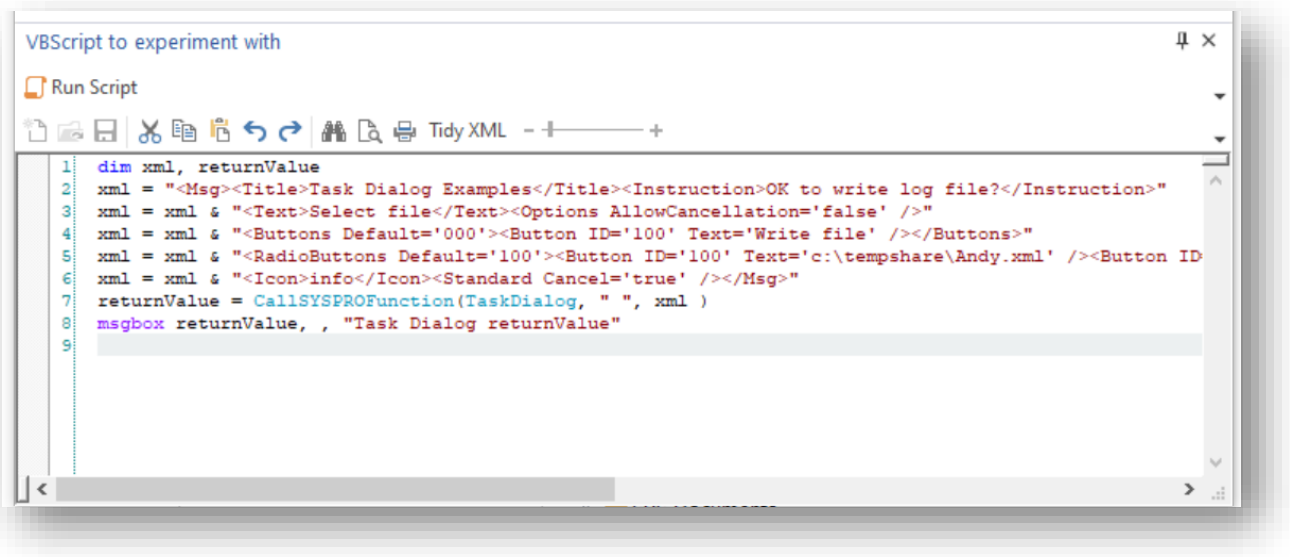
Using the Control

The following code:

```
dim theme
theme ="vbscript"

dim script
script =  "dim xml, returnValue" & vbcrlf
script = script & "xml = """ & tempstr & """" & vbcrlf
script = script & "returnValue = CallSYSPROFunction(TaskDialog, "" "", xml )" & vbcrlf
script = script & "msgbox returnValue, , ""Task Dialog returnValue"""& vbcrlf

call CallSYSPROFunction(SyntaxSetTheme, "SAMTSKS0", theme)

call CallSYSPROFunction(SyntaxInsertText, "SAMTSKS0", script)
```

Generates the following:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



*Please see the sample Application **SAMTSK** for full details.*

XAML Control

The XAML control used in the Application Designer to create a XAML Pane is a lightweight, fast control that uses a subset of the XAML language to create responsive layouts. It enables resizing of elements, graphical representation of data and gives the developer the ability to present information in many ways.

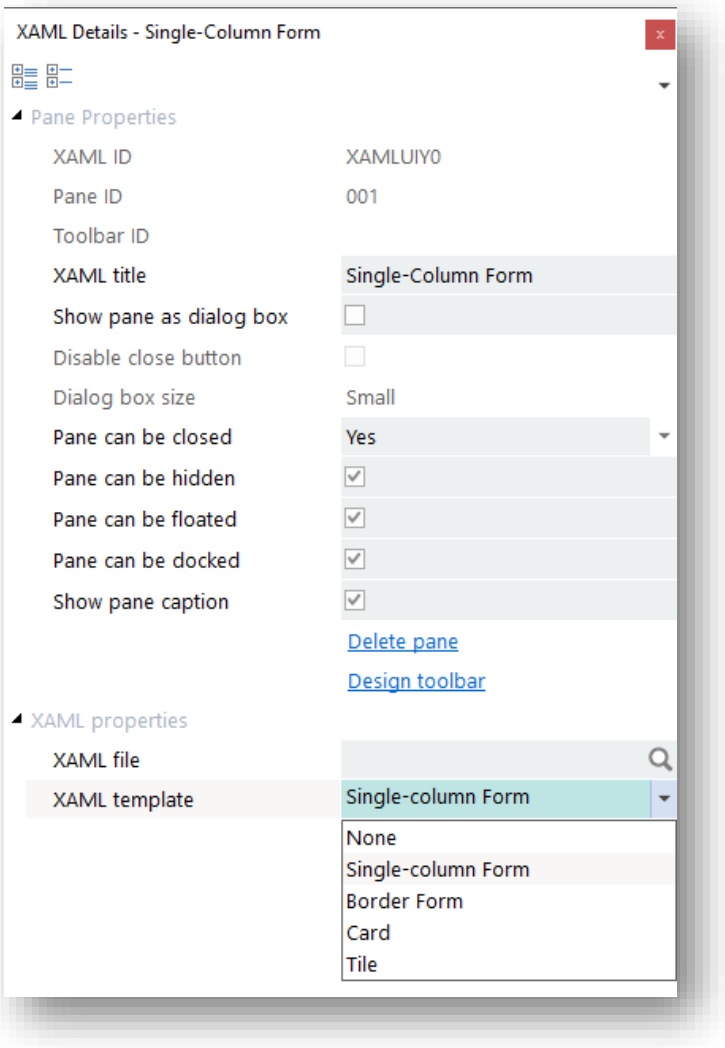
It can be used to create:

- SYSPRO Cards
- Tiles
- Forms
- Free format user interfaces

To simplify the creation of Cards, Tiles, and Forms in XAML, the Application Designer provides templates that can be populated with XML data.

These are Card, Tile, Single-column Form, and Border Form templates:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



The Application Designer uses these templates and an XML string of data to create a XAML page for display. This removes all the complexity of creating the underlying XAML page, grids and styles in the VBScript and allows the developer to concentrate on providing data and content to the pane.

The Application Designer also links up SYSPRO key fields to the standard SYSPRO pop-up menus so that SYSPRO queries can be run from the form.

If no template is used, then the complete XAML Page must be constructed in your code and sent to the XAML control.

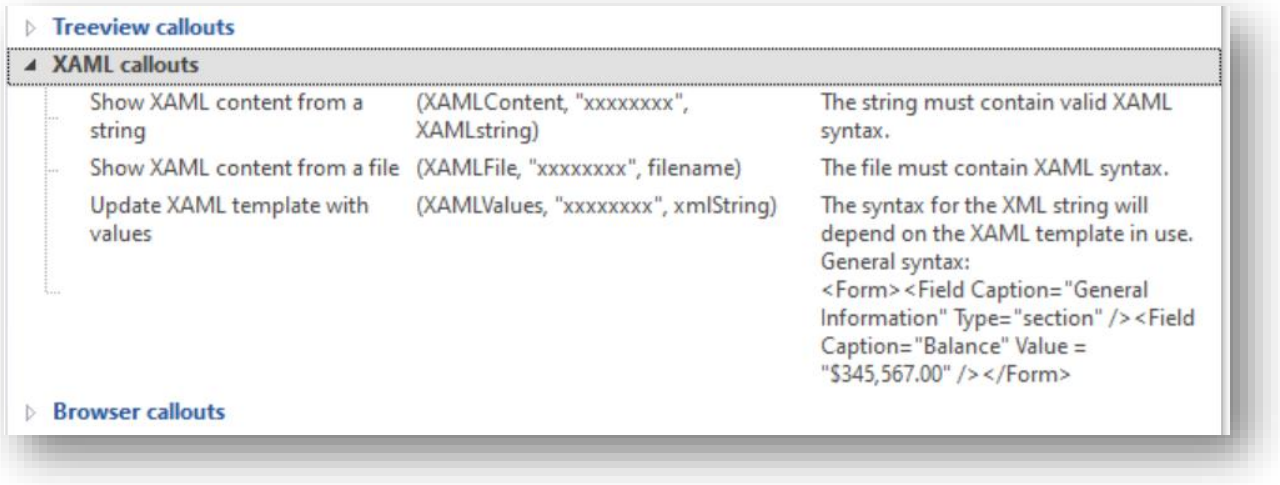
Note that XAML panes differ from traditional control type panes in one major area: The XAML page must be created in its entirety each time any changes are required and the XAML pane refreshed.

Please refer to the sample apps provided with the Application Designer for working examples.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

THE XAML CALLOUTS

The following Callouts are provided for sending XAML data to the XAML control, or XML data to a template for the XAML to be built for you:



Defining the syntax of the XAML Page is crucial. If there are any errors in its construct, then the XAML control will simply not display anything.

Therefore, there are a couple of techniques that can be helpful:

- Use a XAML editor (such as **MICROSOFT EXPRESSION BLEND**) to construct a working prototype of the required result and make sure that what you are creating matches the working model.
  Remember that the Application Designer uses a subset of XAML commands.
  Please see the Supported XAML tags section for more detail.

- Write your XAML to a text file when you create it so that you can copy it into the XAML editor to see: a) if it works, and b) where any errors are.
  There is a VBScript code snippet that can be used to do this.

If the basic XAML page works but occasionally it doesn't display anything, then this is typically caused by the data containing XML reserved characters in it (such as Ampersand (&) or Greater than sign (>)). These often occur in customer name or stock descriptions.

To avoid these, you should run the text through the **`CheckXMLSpecialCharacters`** code snippet which will replace these characters with ones acceptable to XML.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

THE XAML FORM

To simplify the creation of a XAML type form, the Application Designer provides two basic templates.

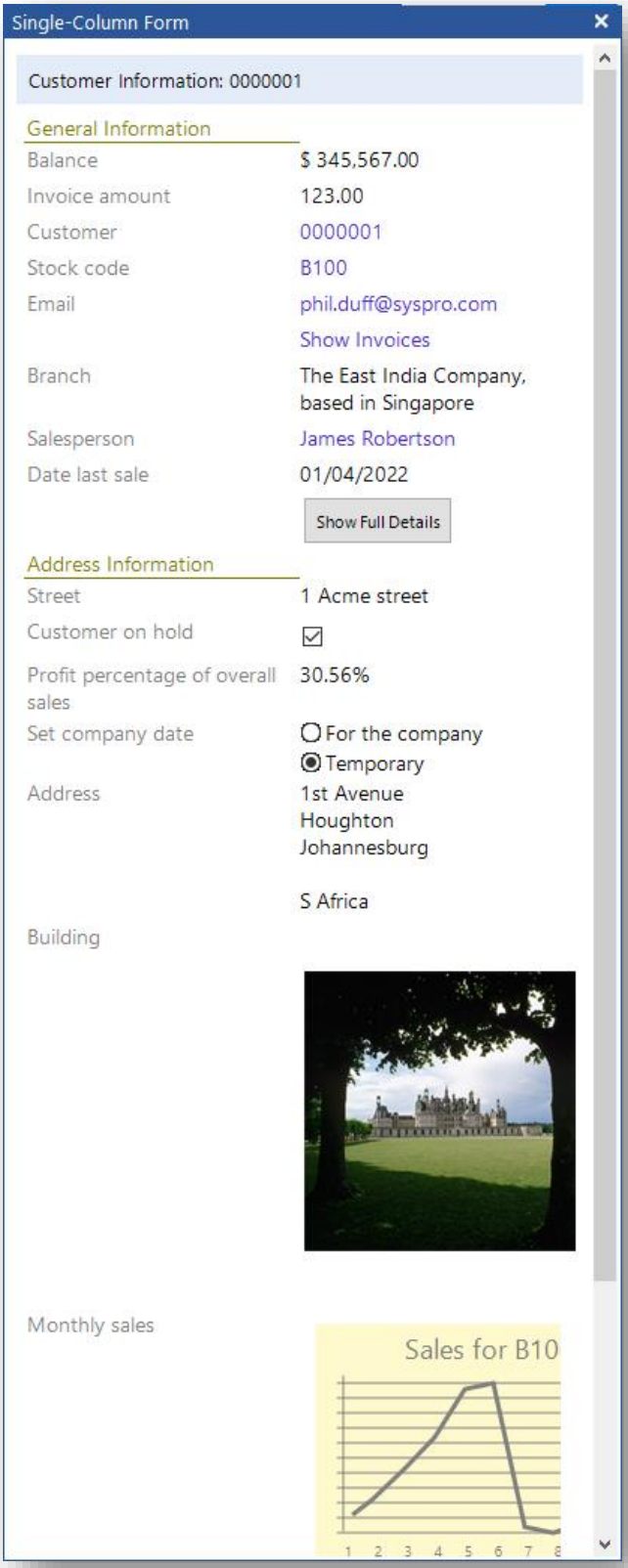
The Single-column Form, which mimics the main SYSPRO style form data panes and the Border Form, which is the same single column structure but with borders around the fields. Future releases of the Application Designer will contain further templates, but the operation will be the same.

Data being sent to the templates is of the following XML format:

```
dim xmlString
' For the CAPTION column, the MinWidth specifies minimum width and the MaxWidth
specifies maximum width.
xmlString = "<Form Heading='Customer Information: 0000001' >" &_
"<Field Caption='General Information' Type='section' />" &_
"<Field Caption='Balance' Value = '$ 345567.00' Type='Number'/>" &_
"<Field Caption='Invoice amount' Value = '123' Type='Number' Decimals='2' />" &_
"<Field Caption='Customer' Value = '1' Tooltip='Acme Brush Company' />" &_
"<Field Caption='Stock code' Value = 'B100' Tooltip='Bicycle'/>" &_
"<Field Caption='Email' Value = 'phil.duff@syspro.com'  Type='hyperlink'/>" &_
"<Field Caption=' ' Value = 'Show Invoices' Tag='TagHyperlink' Type='hyperlink'/>"
&_
"<Field Caption='Branch' Value = 'The East India Company, based in Singapore'/>"
&_
"<Field Caption='Salesperson' Value = 'James Robertson' Tooltip='Based in
Seattle.'/>" &_
"<Field Caption='Date last sale' Value = '2022-04-01' Type='Date'/>" &_
"<Field Caption='Show Full Details' Type='button' Enabled='true'
Tag='TagShowFulldetails' />" &_
"<Field Caption='Address Information' Type='section' />" &_
"<Field Caption='Street' Value = '1 Acme street' />" &_
"<Field Caption='Customer on hold' Value='1' Type='checkbox' Enabled='true'
Tag='TagOnHold' />" &_
"<Field Caption='Profit percentage of overall sales' Value='30.56%' />" &_
"<Field Name='Set company date' Type='4' Value='2' List='For the
company;Temporary;' Tag='TagRadio'/>" &_
"<Field Caption='Address' Value='1st Avenue&#xA;Houghton&#xA;Johannesburg\n\nS
Africa'" +  "/>" &_
"<Field Caption='Building' Image='C:\avis-jpgs\chateau.jpg' Type='image'
Stretch='Uniform' Width='250' Tag='TagImage'/>" &_
"<Field Caption='Monthly sales' Type='linegraph' Title='Sales for B100'
Value='12,23,45,67,100,105,4,0,7,45,50,4' Foreground='gray'
Background='lemonchiffon'  Tag='B100SalesLineGraph' Width='230'/>" &_
"<Field Caption='Monthly sales' Type='columngraph' Title='Sales for B100'
Value='12,23,45,67,100,105,4,0,7,45,50,4' Foreground='white'
Background='limegreen'  Tag='B100SalesBarChart' Width='230'/>" &_
"</Form>"

call CallSYSPROFunction(XAMLValues, "XAMLUIY0", xmlString)
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

Which will produce the following pane:



The visual styling, the type of the data field and what happens when they are clicked on are all controlled by attributes of the Form and Field elements.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

XAML Form – Form and Field attributes

These attributes are explained as follows:

Attribute	Explanation
<Form> attributes	
Heading	This sets the form title.
MaxWidth	This sets the maximum width of the caption column. If the XAML form pane is resized, the caption column will not grow larger than this.
MinWidth	This sets the minimum width of the caption column. If the XAML form pane is resized, the caption column will not shrink smaller than this.
<Field> attributes	
Background	This uses a hex color code or a XAML standard color word to set the background of the Value attribute. The VALUE attribute corresponds to the data field. For graphs, this will set the color of the graph background.
Caption	This indicates the caption for the field. If the caption is recognized as a SYSPRO key field, then it will automatically be hyperlinked and linked to the pop-up menus of standard SYSPRO. To disable this behavior, use the **`Enabled='false'`** attribute *(see below)*. If you don't want a caption next to a control (e.g. Buttons) then set it to a space: **`Caption = ' '`**
CaptionBackGround	This uses a hex color code or a XAML standard color word to set the caption background. Hex values are preceded with a hash sign (e.g. #FF0000 is the hex value for Red). Therefore, **`CaptionBackGround='#FF0000'`** and **`CaptionBackGround='Red'`** produce the same results.
CaptionForeGround	This uses a hex color code or a XAML standard color word to set caption text.
Decimals	This defines the number of decimals for a Number type field.
Enabled	This attribute is set to **`True`** by default as it enables hyperlinks, checkboxes, or buttons. Change this it **`False`** to disable.
Foregound	This uses a hex color code or a XAML standard color word to set value text. For graphs, this will set the color of the graph axis and data.
Image	This indicates a URL to an image file.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

List	This indicates a semi colon list of options for the radio button control.
Stretch	This defines how an image will fit into the image field. Values are: ▪ **Uniform** The content is resized to fit the destination dimensions while it preserves its native aspect ratio. ▪ **UniformToFill** The content is resized to fit the destination dimensions while it preserves its native aspect ratio. If the aspect ratio of the destination rectangle differs from the source, the source content is clipped to fit the destination dimensions. ▪ **Fill** Content is resized to fill the destination dimensions. The aspect ratio is not preserved. ▪ **None** (default) The content preserves its original size.
Tag	This defines a return value if the element is clicked and is used with: ▪ Columngraph ▪ Linegraph ▪ Buttons ▪ Radio buttons ▪ Hyperlinks ▪ Check boxes
Title	This is used to put a title on a graph.
Tooltip	This displays a tooltip when mouse hovers over the caption.
Type	The Type attribute defines the type of field (e.g. Type='Date'). The values are: ▪ **Section** This defines a section. The **CAPTION** is displayed underlined. Any **VALUE** attribute is ignored. ▪ **Date** This will accept a date in the format CCYYMMDD or CCYY/MM/DD or CCYY-MM-DD. The date will be displayed in the same format as core SYSPRO. ▪ **Number** If the **VALUE** is preceded by a currency symbol, then this will be displayed. ▪ Numeric separators (commas and full stops) will be automatically applied. ▪ The number of decimals is defined by the DECIMALS attribute (see above). ▪ If the **DECIMALS** attribute is not set, then the number of decimals is determined from the **VALUE** string itself.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

	▪ Any leading currency sign will be preserved and displayed in front of the number unchanged. ▪ **Button** This defines a button. Button text is set by the CAPTION attribute. When clicked, it raises the **XAMLButton** event with **param1** containing the **TAG** attribute (or the **CAPTION** if the **TAG** attribute has not been explicitly defined). Use the **ENABLED** attribute to enable and disable the button. ▪ **Hyperlink** This defines a hyperlink. Hyperlink text is defined by the **VALUE** attribute. When clicked, it raises a **XAMLHyperlink** event with **param1** containing the **TAG** attribute (or the **CAPTION** if the **TAG** attribute has not been explicitly defined). If the **VALUE** is an email address, then the SYSPRO email interface is launched. For an email the event is not subsequently passed to the VBScript. ▪ **Image** This defines an image field. The image size is controlled by **STRETCH** and **WIDTH** attributes. When clicked, it raises the **XAMLHyperlink** event with **param1** containing the **TAG** attribute (or the **CAPTION** if the **TAG** attribute has not been explicitly defined). ▪ **Checkbox** This defines a checkbox. The check status is set by the **VALUE** attribute. 0 = unchecked, 1 = checked. When clicked, it raises the **XAMLCheckBox** event with **param1** containing the **TAG** attribute (or the **CAPTION** if the **TAG** attribute has not been explicitly defined) and **param2** holds the check status; **0** for unchecked and **1** for checked. ▪ **Radio** This defines a radio button. Options are defined in a **LIST** attribute (see above). The radio button options' position in the list are numbered from 1. A default option can be set by the **VALUE** attribute and a return value is defined in the **TAG** attribute. When an option is clicked, it raises the **XAMLRadioButton** event. The return value in **param1** contains the **TAG** attribute (or the **CAPTION** if the **TAG** attribute has not been explicitly defined). **Param 2** contains the position of the option in the list.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

	▪ **Linegraph** This displays a line graph. Its values come from the Value attribute which should be a comma separated list of up to a maximum of 12 values. ▪ **Columngraph** This displays a column graph. Its values come from the **VALUE** attribute which should be a comma separated list of up to a maximum of 12 values. ▪ **For graphs** The x-axis labels are shown as 1,2,3,4,5,6,7,8,9,10,11,12, but can be overridden with the **AxisLabels** attribute which has the labels separated (e.g. **AxisLabels='Jan,Feb,Apr** ... '). These labels should not be longer than 4 or 5 characters. When a graph is clicked, it raises the **XAMLButton** event with **param1** containing the **TAG** attribute (or the **CAPTION** if the **TAG** attribute has not been explicitly defined).
Width	This defines the width of an image or a graph. The width and height of these controls is always the same. If not set, then the default is **100**.
Value	This indicates the value to be displayed next to the caption. This is the data field displayed in the form. To insert a newline in a string, use the XAML syntax for a newline **
** or use a **\n** character.

CARDS

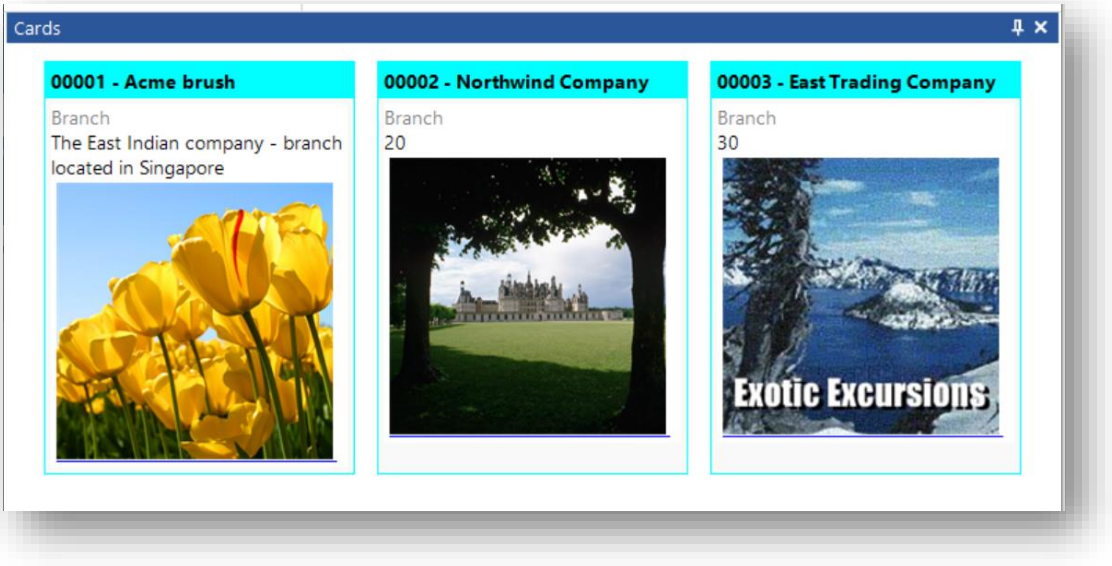
The card template operates in a similar manner to Forms except that a new card is created for each **<Field>** element.

The following code:

```
dim xmlString
xmlString = "<Form>" &_
"<Field Heading='00001 - Acme brush' Caption='Branch' Value = 'The East
Indian company - branch located in Singapore' Image='C:\avis-
jpgs\tulips.jpg' Tag='Tultips'/>" &_
"<Field Heading='00002 - Northwind Company' Caption='Branch' Value = '20'
Image='C:\avis-jpgs\chateau.jpg' Tag='Chateau'/>" &_
"<Field Heading='00003 - East Trading Company' Caption='Branch' Value =
'30' Image='C:\avis-jpgs\bayshore.jpg' Tag='Bayshore'/>" &_
"</Form>"
call CallSYSPROFunction(XAMLValues, "XAMLUIY1", xmlString)
```

Generates three cards as follows:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



The pane is responsive, so the cards will move to accommodate the new size as the pane is resized.

CARDS – FIELD ATTRIBUTES

Attribute	Explanation
Heading	A heading at the top of the card.
Caption	A caption on the first line of the card.
Value	A data element displayed under the caption.
Image	A URL to an image.
Tag	Return value when image clicked.

When the image is clicked, it raises the **`XAMLHyperlink`** event and **`Param1`** will contain the value of **`TAG`**. If **`TAG`** is not defined, then param1 will be blank.

TILES

The tile template operates in the same way as a card. A single tile is in essence a giant button. Each **`<Field>`** element creates a new tile, and the pane is responsive enabling the tiles to re-organize to fit the available space.

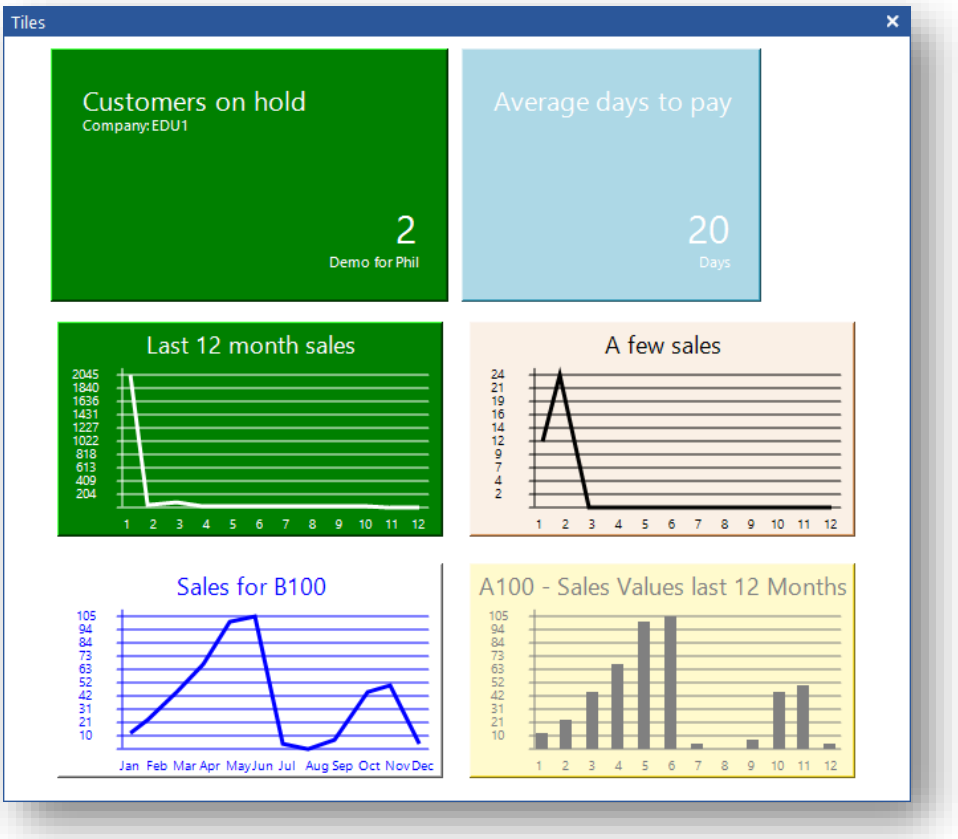
Note: This TILE template is not to be confused with the Tiles that can be shown in a Tile Layout in an Avanti Web View.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

The following code:

```
dim xmlstring
xmlstring = "<Form>" &_
"<Field Title='Customers on hold' Subtitle='Company: EDU1' Value='2'
Footer='Demo for Phil' Foreground='White' Background='Green' Width='300'
Tag='On Hold'" &_
" Tooltip='&lt;Border&gt;&lt;Image Source=""file://c:\avis-jpgs\tulips.jpg""
Width=""200"" Height=""200""
Stretch=""UniformToFill""/&gt;&lt;/Border&gt;'/>" &_
"<Field Title='Average days to pay' Subtitle=' ' Value='20' Footer='Days'
Foreground='White' Background='LightBlue' Width='auto' Tag='Average Days'/>"
&_
"<Field Type='linegraph' Title='Last 12 month sales' Subtitle=' '
Value='2045,25,67,4,6,7,8,9,10,6' Foreground='White' Background='Green'
Width='auto' Tag='Sales by Month'/>" &_
"<Field Type='linegraph' Title='A few sales' Subtitle=' ' Value='12,24'
Foreground='Black' Background='Linen'  Tag='FewSales' Tooltip='Click to see
details'/>" &_
"<Field Type='linegraph' Title='Sales for B100'
Value='12,23,45,67,100,105,4,0,7,45,50,4' Foreground='Blue'
Background='White'  Tag='B100Sales'
AxisLabels='Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec'/>" &_
"<Field Type='columngraph' Title='A100 - Sales Values last 12 Months'
Value='12,23,45,67,100,105,4,0,7,45,50,4' Foreground='gray'
Background='lemonchiffon'  Tag='A100' />" &_
"</Form>"
call CallSYSPROFunction(XAMLValues, "XAMLUIY2", xmlString)
```

Generates the following tiles:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

TILE – FIELD ATTRIBUTES

Attribute	Explanation
Title	The main heading at the top of the Tile.
Subtitle	A second (smaller) description below the title.
Value	A data element(s) depending on Type attribute (see below).
Footer	This is the text displayed at the bottom of the tile.
Foreground	This uses a hex color code or a XAML standard color word to set the tile foreground (which is the text color of any titles, subtitles, footers, and values; or the color of graph lines and labels). Hex values are preceded with a hash sign (e.g. **#FF0000** is the hex value for Red). *For example:* **Foreground='#FF0000' and Foreground='Red'** produce the same results.
Background	This uses a hex color code or a XAML standard color word to set the tile background.
Width	This specifies the width of the tile. **Width='Auto'** will size the tile to fit the data.
Type	This defines the type of control hosted by the tile: ▪ **Linegraph** This displays a line graph. Its values come from the **Value** attribute which should be a comma separated list of up to a maximum of 12 values. ▪ **Columngraph** This displays a column graph. Its values come from the **VALUE** attribute which should be a comma separated list of up to a maximum of 12 values. ▪ **For graphs** The x-axis labels are shown as 1,2,3,4,5,6,7,8,9,10,11,12, but can be overridden with the **AxisLabels** attribute which has the labels separated (e.g. **AxisLabels='Jan,Feb,Apr ... '**). These labels should not be longer than 4 or 5 characters. When a graph is clicked, it raises the **XAMLButton** event with **Param1** containing the **TAG** attribute (or the **CAPTION** if the **TAG** attribute has not been explicitly defined).
Tag	This is the return value when the tile is clicked.

Clicking on a tile will raise the **XAMLButton** event and the value of the **TAG** attribute is returned in **Param1**.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

OTHER USES FOR XAML

XAML can be used in several controls, including:

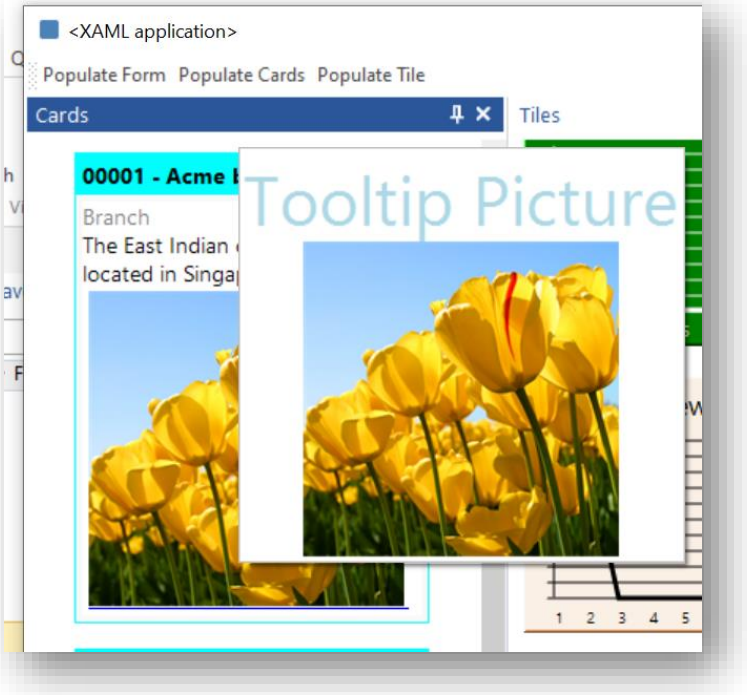
- Pane tooltips can be set by sending a string of XAML to a pane.
- Tooltips on card images can be set by providing XAML in the tooltip attribute.
- XAML can be used in tool tips for panes and tiles.
- It can also be used for a Tree view item.
- The XAML is processed in a grid control, so if the XAML you use works in a Grid then it will work in a tooltip.

XAML AND PANE TOOLTIPS

This code sets the tooltip for a pane to be an image:

```
call PopulateCard()
Tooltip = "<Border><StackPanel>" &_
          "<TextBlock Foreground='LightBlue' FontSize='43' FontFamily='Segoe UI' Text='Tooltip Picture'/>" &_
          "<Image Source='file://c:\avis-jpgs\tulips.jpg' Width='200' Height='200' Stretch='UniformToFill'/>" &_
          "</StackPanel></Border>"
call CallSYSPROFunction(PaneChangeTooltip, "XAMLUIY1", Tooltip)
```

Hover the mouse over the pane title bar and the tooltip shows:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

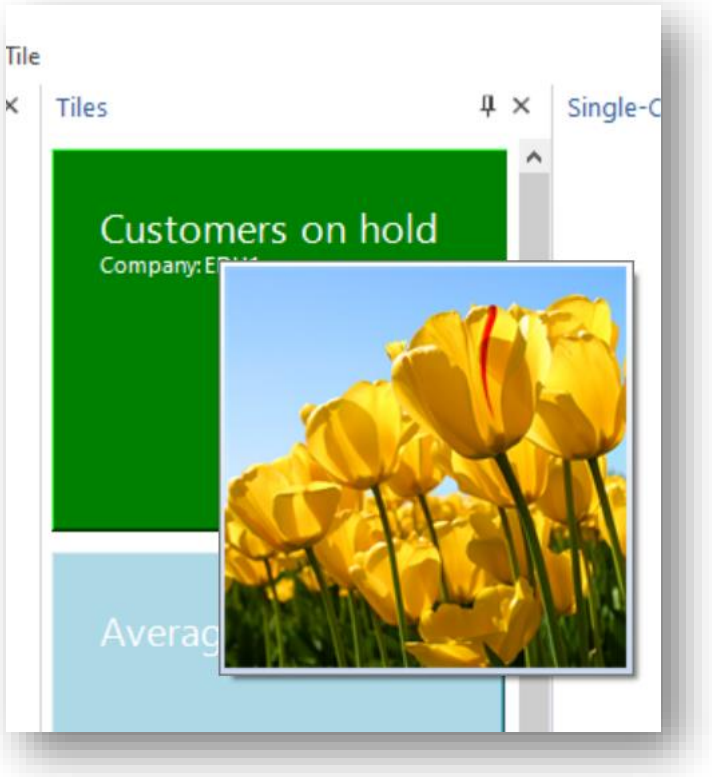
XAML and Card tooltips

The code below creates a tooltip on a tile:

```
& "<Field Title='Customers on hold' Subtitle='Company: EDU1' Value='2' Footer='Demo for Phil' " _
& "Foreground='White' Background='Green' Width='300' Tag='On Hold' " _
& "Tooltip='&lt;Border&gt;&lt;Image Source=""file://c:\avis-jpgs\tulips.jpg"" " _
& "Width=""200"" Height=""200"" Stretch=""UniformToFill""/&gt;&lt;/Border&gt;'/>" _
```

Note the **<** and **>** characters have been replaced with XML acceptable codes **`<`** and **`>`**.

This could be used to show more information for the tile.



Handling XAML strings

If provided as a string to an attribute for a control, you need to make sure that any XML reserved characters are replaced with their relevant encoding, otherwise XAML and XML get mixed up and nothing will work.

There is a code snippet provided in the Application Designer called **`CheckXMLSpecialCharacters`** which will parse a string and substitute the relevant codes if required and of course you can hard code the string.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

SUPPORTED XAML TAGS

The following is a list of supported XAML tags and their attributes:

XAML tag	Attributes	
Page	▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width	▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ MaxHeight
Border	▪ Cursor ▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width ▪ Height	▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ MaxHeight ▪ Padding ▪ Background ▪ BorderThickness ▪ BorderBrush
TextBlock	▪ Cursor ▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width ▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ MaxHeight	▪ Padding ▪ Background ▪ Foreground ▪ TextWrapping ▪ TextAlignment ▪ TextDecorations ▪ FontSize ▪ FontWeight ▪ FontFamily ▪ FontStyle ▪ BaselineAlignment ▪ Text
Image	▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width	▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ MaxHeight ▪ Source
Run	▪ Cursor ▪ Background ▪ Foreground ▪ TextDecorations ▪ FontSize	▪ FontWeight ▪ FontFamily ▪ FontStyle ▪ BaselineAlignment ▪ Text
LineBreak		

© SYSPRO. All Rights Reserved. All trademarks are recognized.

XAML tag	Attributes	
Span	▪ Background ▪ Foreground ▪ TextDecorations ▪ FontSize	▪ FontWeight ▪ FontFamily ▪ FontStyle
Bold	▪ Background ▪ Foreground ▪ TextDecorations ▪ FontSize	▪ FontWeight ▪ FontFamily ▪ FontStyle
Italic	▪ Background ▪ Foreground ▪ TextDecorations ▪ FontSize	▪ FontWeight ▪ FontFamily ▪ FontStyle
Underline	▪ Background ▪ Foreground ▪ TextDecorations ▪ FontSize	▪ FontWeight ▪ FontFamily ▪ FontStyle
InlineUIContainer	▪ Background ▪ Foreground ▪ TextDecorations ▪ FontSize	▪ FontWeight ▪ FontFamily ▪ FontStyle
Hyperlink	▪ Background ▪ Foreground ▪ TextDecorations ▪ FontSize	▪ FontWeight ▪ FontFamily ▪ FontStyle
Panel	▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width	▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ MaxHeight ▪ Background
StackPanel	▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width ▪ Height	▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ MaxHeight ▪ Background ▪ Orientation

© SYSPRO. All Rights Reserved. All trademarks are recognized.

XAML tag	Attributes	
WrapPanel	▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width ▪ Height ▪ MinWidth	▪ MaxWidth ▪ MinHeight ▪ MaxHeight ▪ Background ▪ Orientation ▪ ItemHeight ▪ ItemWidth
Canvas	▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width ▪ Height ▪ MinWidth	▪ MaxWidth ▪ MinHeight ▪ MaxHeight ▪ Background ▪ Left ▪ Right ▪ Top ▪ Bottom
Grid	▪ Style ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width ▪ Height	▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ MaxHeight ▪ Background ▪ ColumnDefinitions ▪ RowDefinitions
ColumnDefinition	▪ Width ▪ MinWidth	▪ MaxWidth
RowDefinition	▪ Height ▪ MinHeight	▪ MaxHeight
Polygon	▪ Width ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width	▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ Points
Polyline	▪ Width ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width	▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ Points

© SYSPRO. All Rights Reserved. All trademarks are recognized.

XAML tag	Attributes	
Rectangle	▪ IsMouseOver ▪ Width ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment	▪ Width ▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight
Ellipse	▪ Width ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment	▪ Width ▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight
Line	▪ Width ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment ▪ Width	▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ X1, Y1, X2, Y2
Button	▪ IsPressed ▪ Style ▪ Width ▪ Margin ▪ ClipToBound ▪ HorizontalAlignment ▪ VerticalAlignment	▪ Width ▪ Height ▪ MinWidth ▪ MaxWidth ▪ MinHeight ▪ Content
CheckBox		
RadioButton		
ScrollViewer	▪ Style ▪ VerticalScrollBarVisibility	▪ HorizontalScrollBarVisibility
SolidColorBrush	▪ Color	
LinearGradientBrush	▪ StartPoint ▪ EndPoint	▪ GradientStops
GradientStop	▪ Color	▪ Offset
Style	▪ TargetType ▪ BasedOn	▪ Resources
Setter	▪ Property	▪ Value
Events	▪ MouseEnter ▪ MouseLeave ▪ MouseMove ▪ MouseLeftButtonUp	▪ MouseLeftButtonDown ▪ MouseRightButtonUp ▪ MouseRightButtonDown

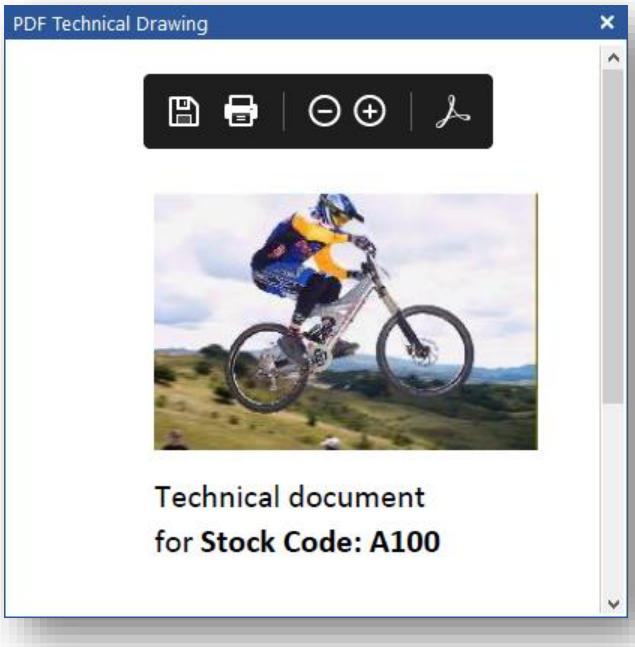
© SYSPRO. All Rights Reserved. All trademarks are recognized.

Document Viewer

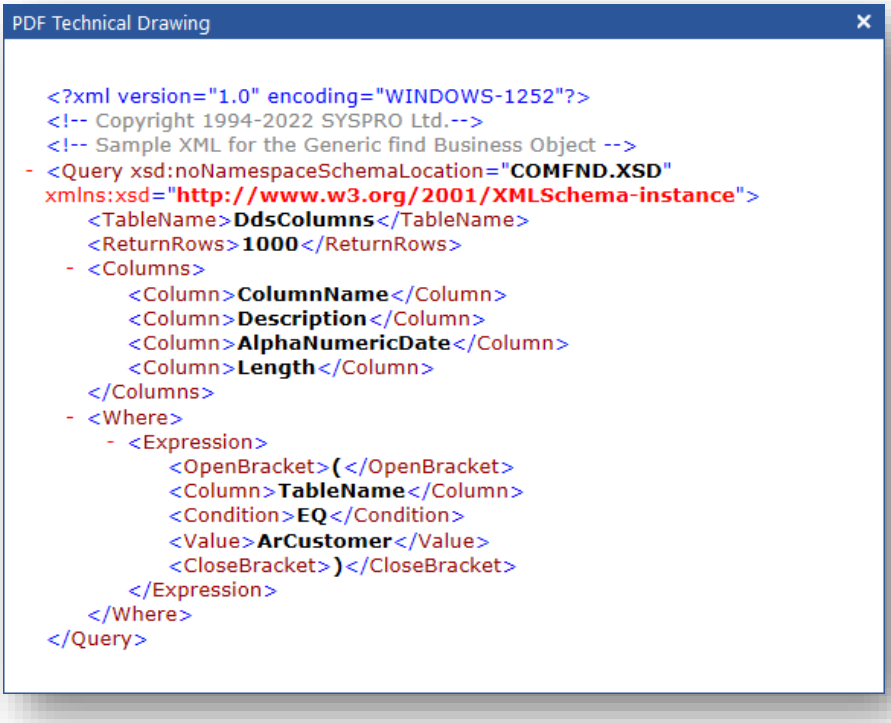
The document viewer control allows you to view certain types of documents (PDF, Text, XML) in the viewer.

For example:

If you specify a PDF document to view, then the document will be rendered:

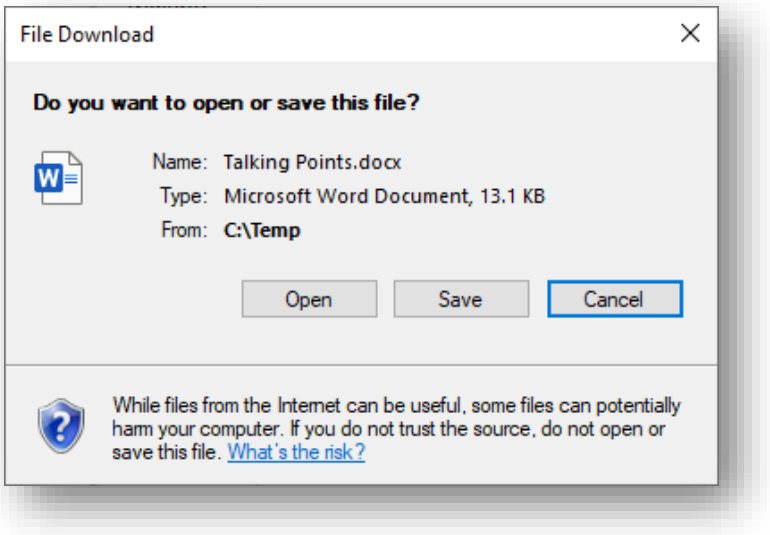


Similarly, pointing the viewer to an XML file will render it like this:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

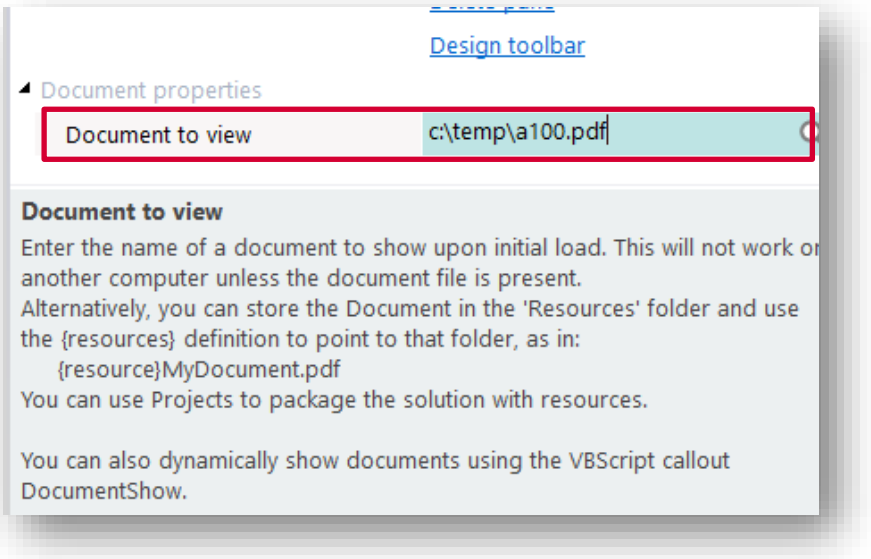
Whereas pointing the viewer to a Word document (which is not natively understood by the viewer control) will result in this message box being shown:



DEFINING THE DOCUMENT TO VIEW

There are two ways of specifying the name of a document to view:

1. In the document properties:



2. Using the **DocumentShow** VBScript callout to dynamically change the content:

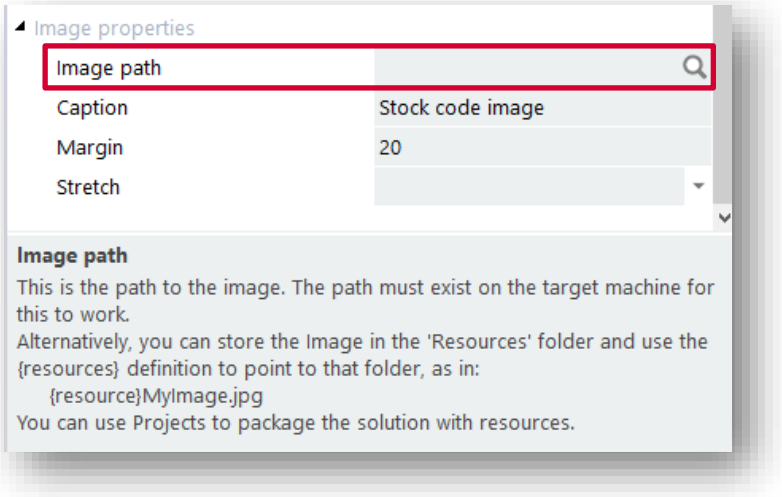
```
dim filename
filename = "C:\Temp\Demo Documents\a100.pdf"
call CallSYSPROFunction(DocumentShow, "NOTE00P0", filename)
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

IMAGE VIEWER

There are two ways of displaying images in the control:

1. By specifying a path to the image in the properties pane:



2. Or by using the VBScript callout **ImageRefresh** to dynamically change the image:

```
call CallSYSPROFunction(ImageRefresh, "NOTE00I0", "c:\avis-jpgs\chateau.jpg")
```

Optionally, you can also specify a caption - either in the **Properties** window or dynamically using the **ImageCaption** callout:

```
call CallSYSPROFunction(ImageCaption, "NOTE00I0", "new image caption")
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

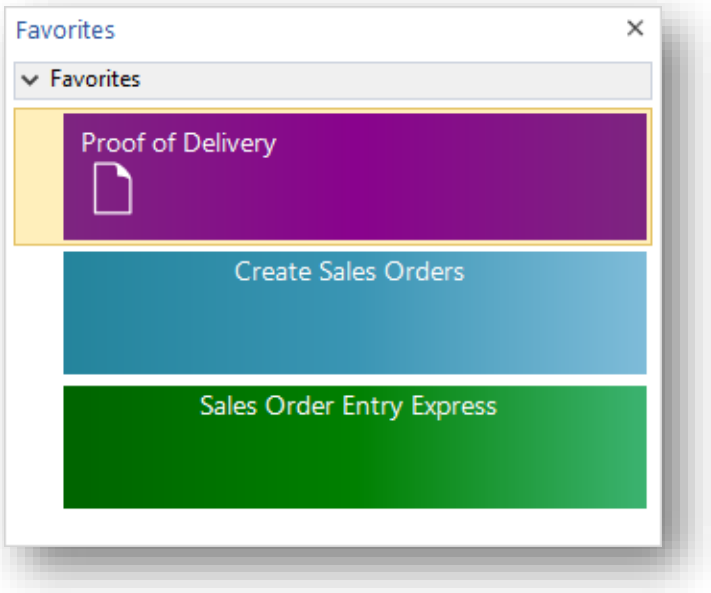
Tile Control

The Tile control is the same control that is used for the **Favourites** pane in SYSPRO's main menu and is primarily used for navigation content.

Each tile can contain content (a title, a tooltip, and an image) with various attributes (style, colors, icons) and can launch a SYSPRO program or executable.

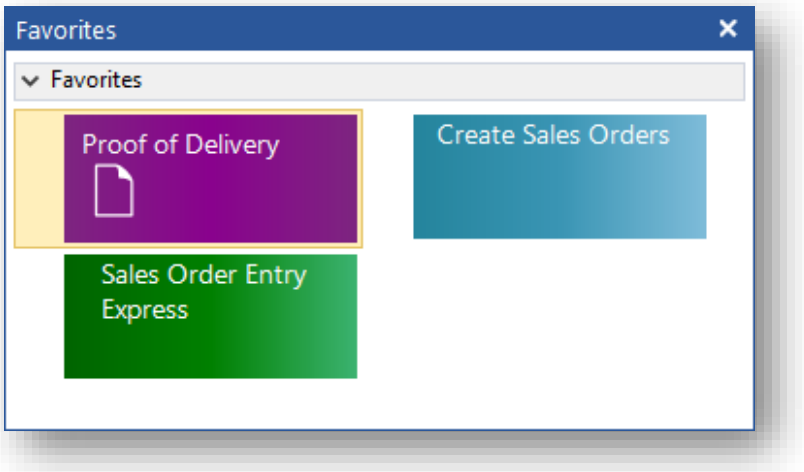
Tiles can also be grouped into **Categories**.

Here's an example of some tiles:



Tiles can be stretched across the control area (as in the above example) or can be displayed in multiple columns.

Here's an example of the same tiles but with **Multi-column groups** switched on.

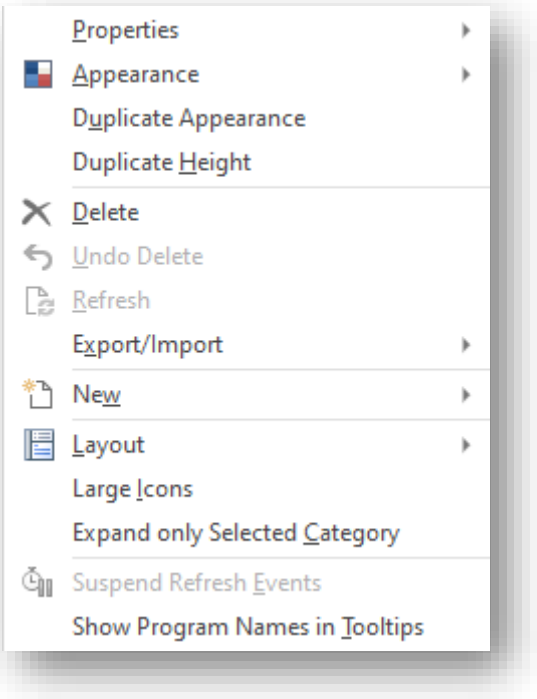


© SYSPRO. All Rights Reserved. All trademarks are recognized.

Designing Tiles

To add a tile to the control, proceed as follows:

1. Drag a program from the **Programs List** or **Recent Programs** list in the main menu and drop it on the Tile control.
2. Drag a tile from your **Favourites** menu items onto the Tile control.
3. Drag any file from Windows Explorer onto the Tile control.
4. Right-click on the Tile control and a popup menu will appear:



From here you can create a new Tile, delete a tile, add a category, and change various styling options.

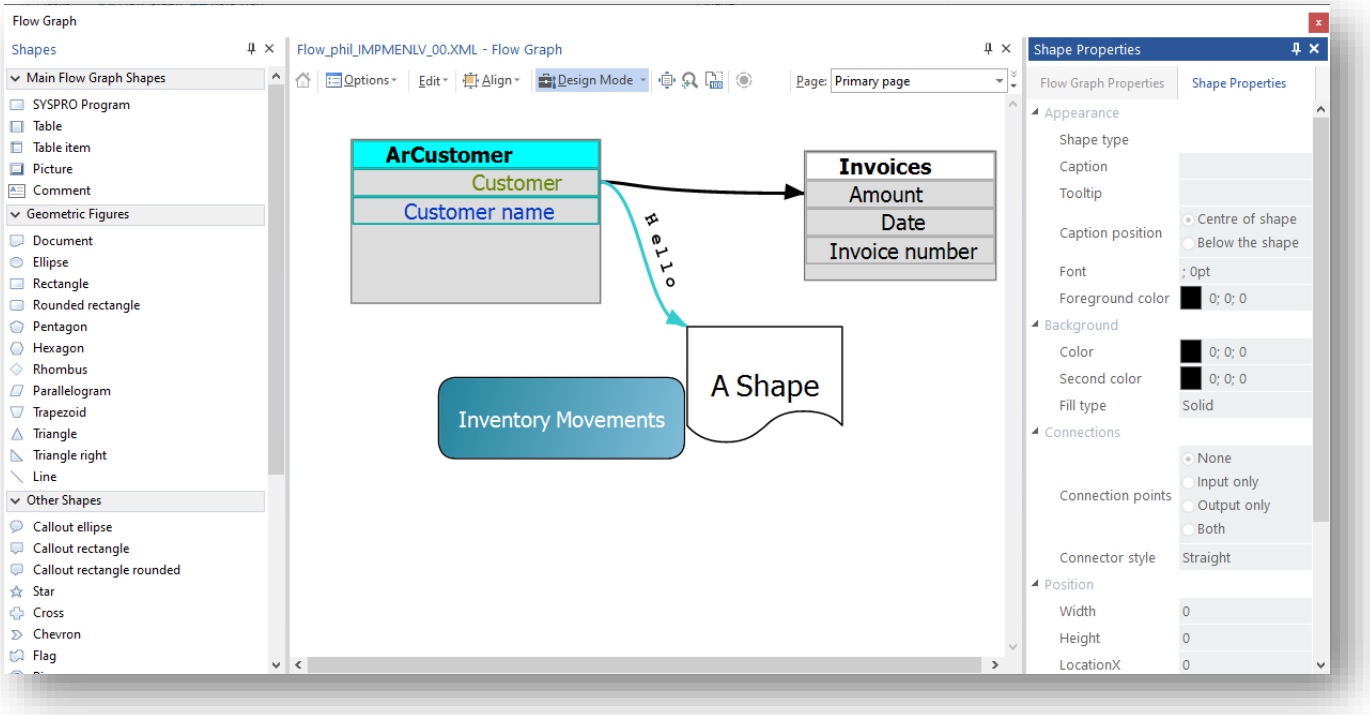
Flowgraph Control

The flowgraph control can view any existing Flowgraph (that was created in the Flowgraph designer) found in the SYSPRO main menu.

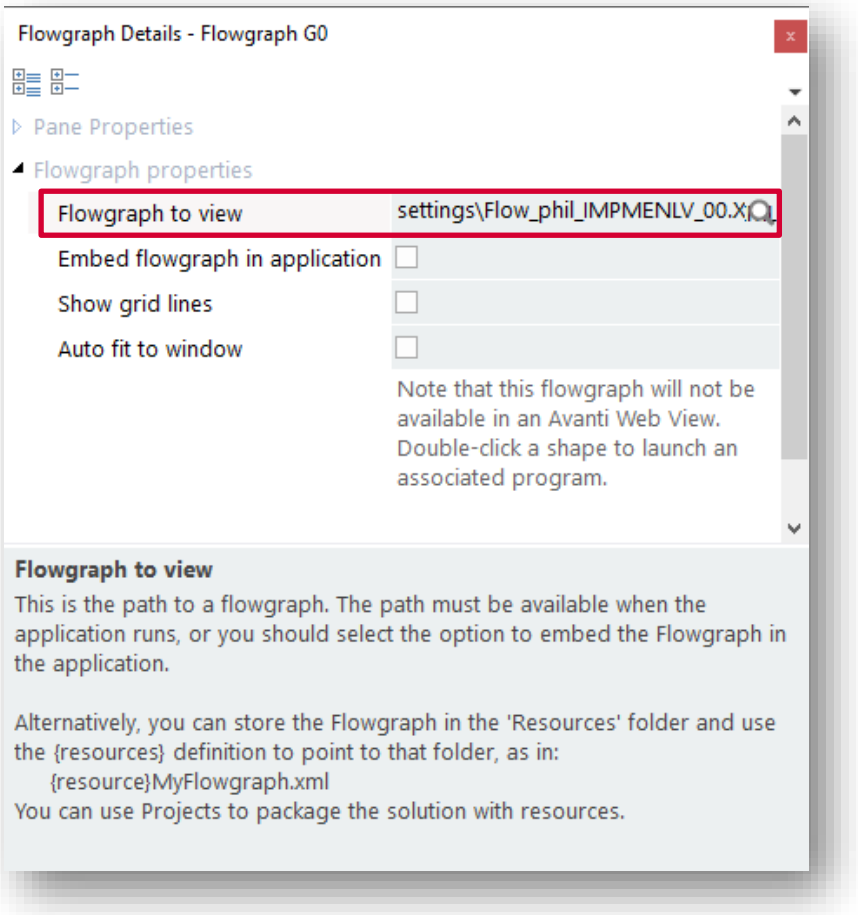
Flowgraphs are a great way of showing processes to a user, especially since clicking on a 'shape' can launch any SYSPRO program or executable.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

You should use the SYSPRO Flowgraph designer to create your flowchart:

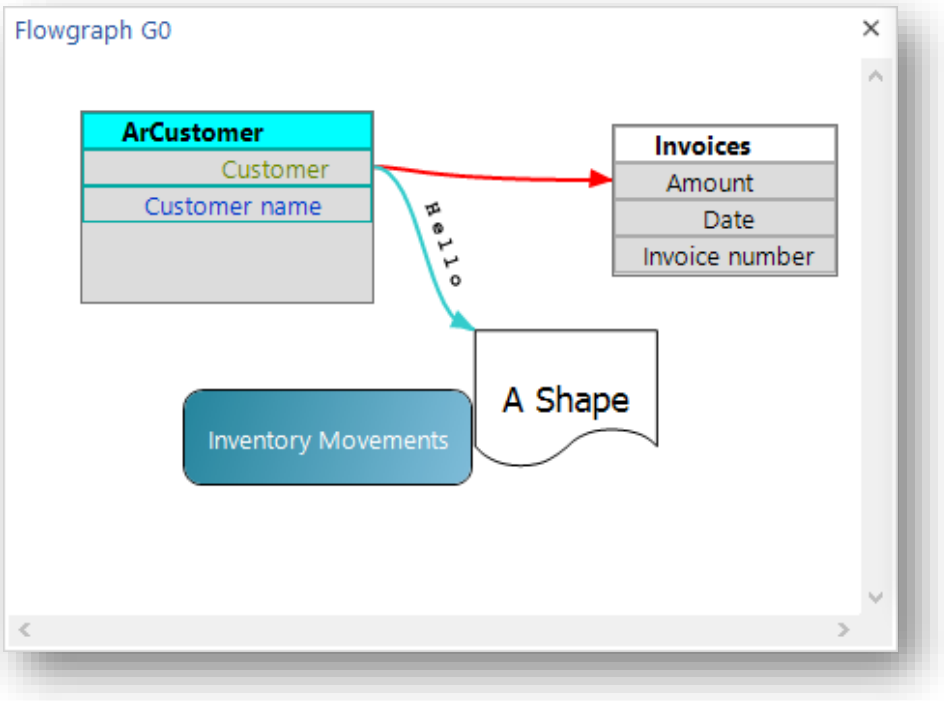


and then specify the path to the Flowgraph in the **Flowgraph to view** property:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

The flowgraph will then be displayed in the control:



You should consider enabling the **Embed flowgraph in the application** option, as this option ensures that whatever flowgraph is specified, the contents are embedded in the application text file when the application is saved.

You can then deploy the application to the target machine knowing that the Flowgraph will work on that machine.

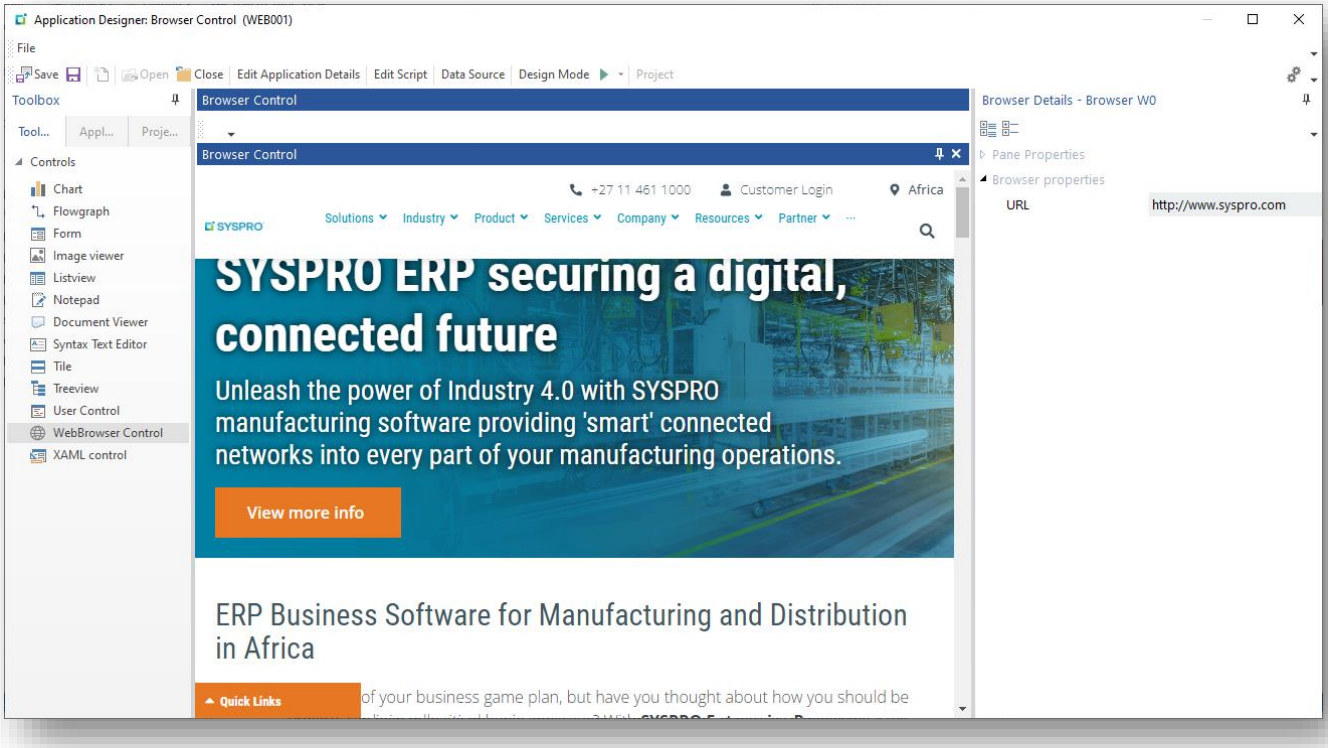
If you do not embed the flowgraph in the application, the flowgraph path must exist on the target machine.

Note: This flowgraph control cannot be used in Avanti.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

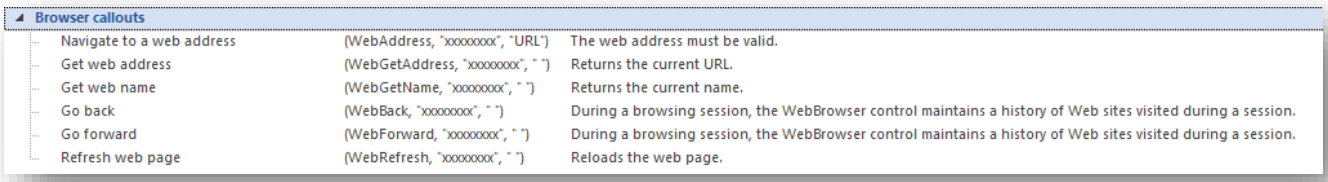
WEBBROWSER CONTROL

The browser control can navigate to any specified URL:



Or you can use VBScript callouts to dynamically navigate to any URL.

Here are the available callouts:



Note: The web browser control will use the latest **MICROSOFT** browser technology that's available on the computer.

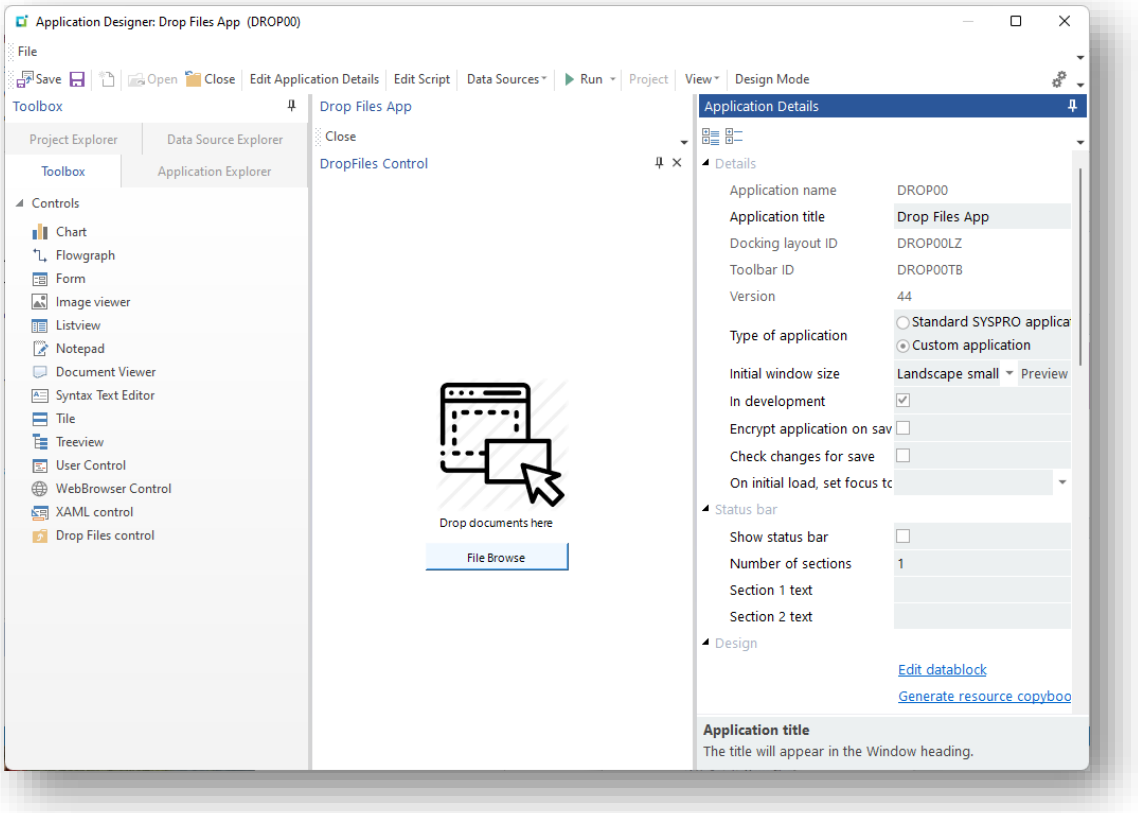
If **MICROSOFT EDGE WEBVIEW2** has been installed, for example, then that will be used to render content.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

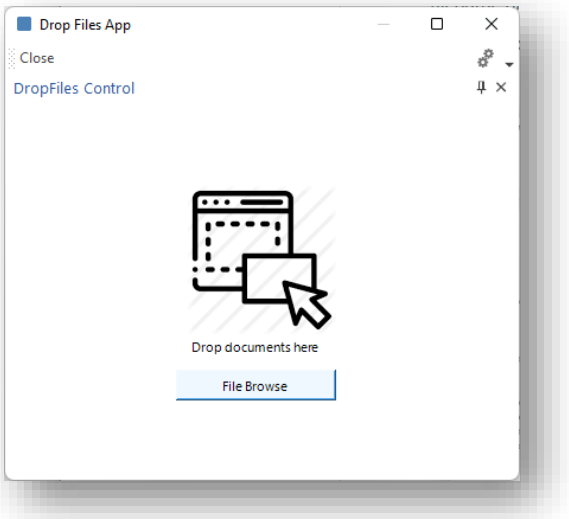
Drop Files Control

This control allows files to be dragged from Windows Explorer onto a document control area. You can use this control to manage documents and files that your application needs to process. This might alleviate the need to have a file browse button on a form to process a filename one at a time.

In your application, click on **Drop Files control** and a pane will be added to your application like this:

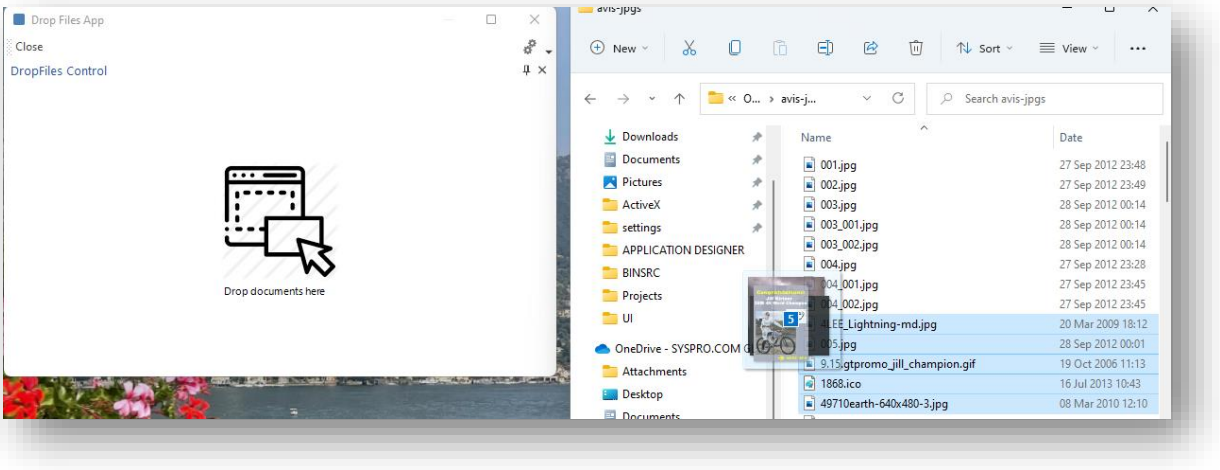


Now run the application which will look like this:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Now select and drag one or more files from Windows Explorer onto the drop area of the control:



(alternatively, you can click on **File Browse** to select files by browsing the file system).

When one or more files are dragged and dropped onto the control area the script event **DroppedFilesEvent** will be raised in the **AppDesigner_OnUserEvent**. The list of files dropped onto the control will passed into **Param1**, each filename separated by a TAB character.

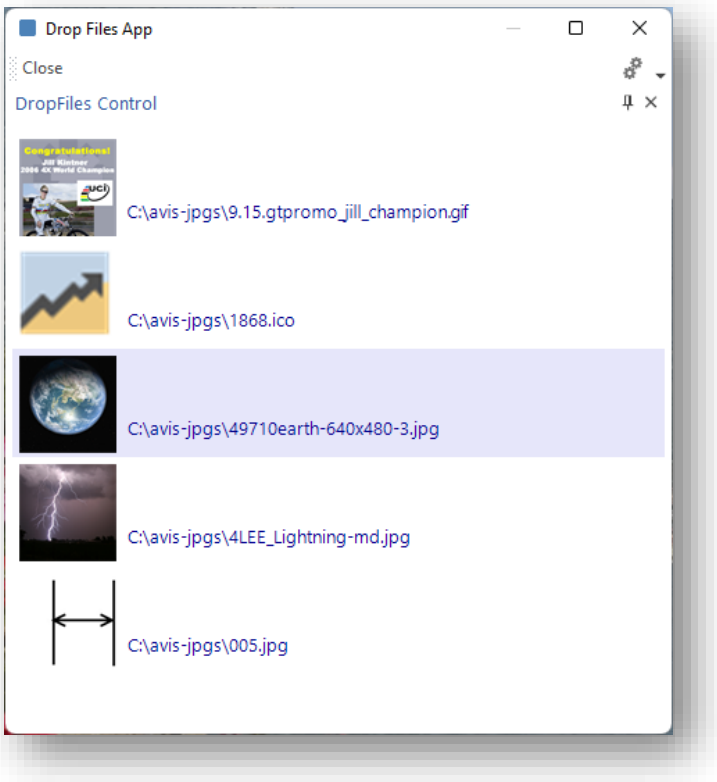
Here's some example code to process the list of dropped files:

```
' Dropped files event. Param1 contains a list of filenames, each separated by
a TAB character
Function
AppDesigner_OnUserEvent(EventType,EventName,EventID,Param1,Param2,Param3)
if eventType = DroppedFilesEvent and eventID = DroppedFiles then
   dim FileNameArray, x1, ArraySize
   FileNameArray = split(Param1, Chr(10))
   ArraySize = ubound(FileNameArray)
   for x1 = 0 to ArraySize
       msgbox "Filename: " + FileNameArray(x1)
   next
end if
```

Optionally, you can also show the list of dropped files in the control area.

When you click on any of the listed files, a XAML hyperlink event is triggered and **Param1** will contain the name of the file:

© SYSPRO. All Rights Reserved. All trademarks are recognized.



You can also change the **Number of columns to show** in this list.

This is what the list will look like with 2 columns to show:

Finally, you can change the Image size to be Normal, Medium or Large. This property applies to files that are of type image.

The list of files can be cleared by clicking on the **Clear List** button.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Working with Projects

Projects can help manage multiple applications and resources, especially if deploying a full solution to an end-user. Projects can also be useful simply to keep a track of applications that you are working on.

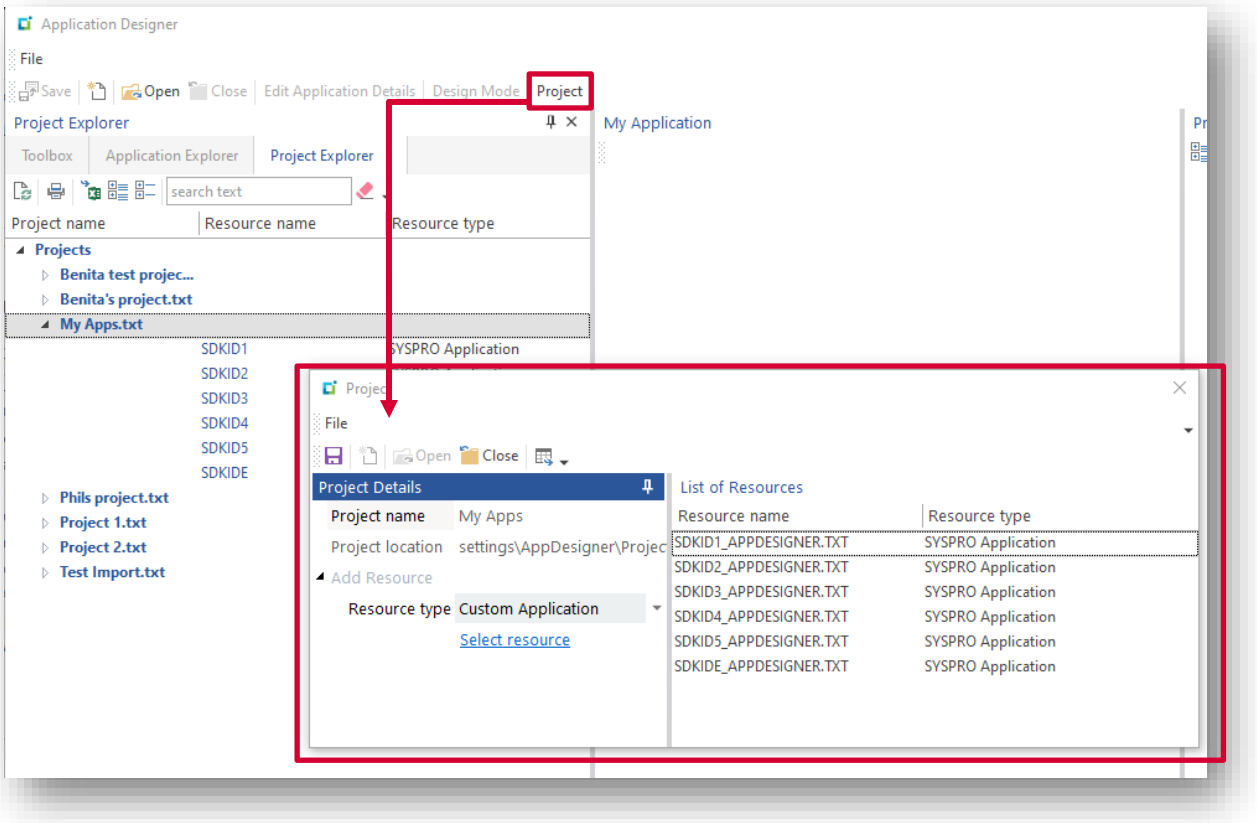
Projects can contain resources including:

- Custom Applications
- Standard SYSPRO applications
- Web Views
- .NET User controls
- VBS Modules

Additionally, projects can include 'Other resources'. These can be any type of file in any folder location that can be added to the project. When importing such files, they will always be imported to the `...\settings\AppDesigner\Resources` folder on the target machine.

Managing Projects

To manage Projects, click on the **Project** button in the Application Designer toolbar:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Creating New Projects

To create a new project, click on the **NEW** button and enter a project name.

- The name can be anything up to 40 characters long but cannot contain any special character, such as `/ ? : & \ * " < > | # or %`.
- The project will be saved in the `...\settings\AppDesigner\Projects` folder.

You can add 'resources' to your project by first selecting the **Resource type** and then clicking **Select resource** hyperlink.

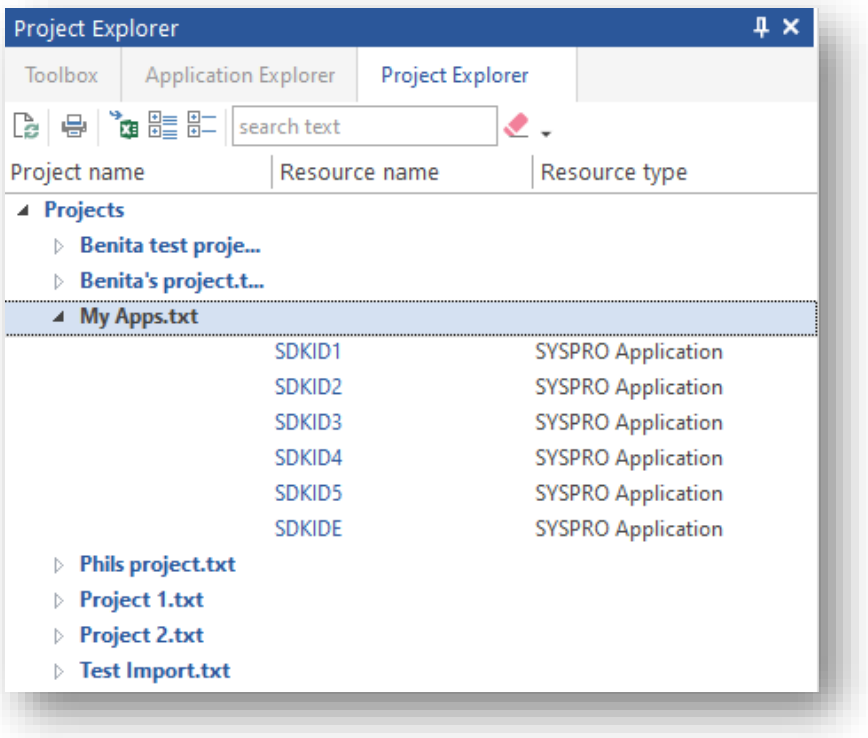
- The selected resources are shown in the **List of Resources** grid.
- You can delete a resource in the grid by pressing the DEL key.

You can also reposition any resource item by dragging it up or down the grid.

- You might wish to do this as when the project is exported and then imported on another machine, the list will be shown exactly in the same order.
- You might wish to add a 'other resource' to your project which contains installation or deployment instructions to the end-user, and you want that resource to appear at the top of the list.

After clicking SAVE, the resource list is saved in the projects named file. You can view this list either within the Projects window by clicking the OPEN button and selecting the appropriate project, or by using the Project Explorer.

The Project Explorer displays all the projects on the local machine:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

Maintaining Projects

You can click on the Project name to edit the project directly, or you can click on any application to edit it directly. This makes it easy to manage a group of applications, both from an editing perspective and from an export perspective.

Exporting a Project

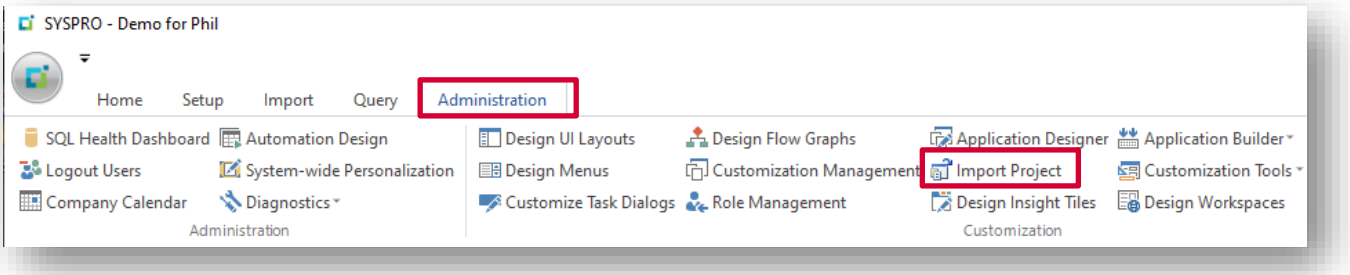
When editing a Project, you can click on the **Export** button.

- This will ZIP up the resource files and place the ZIPPED file in your `…\AppDesigner\Projects` folder.
- The name of this zip file will be the name of the project with the suffix `.prjzip`.

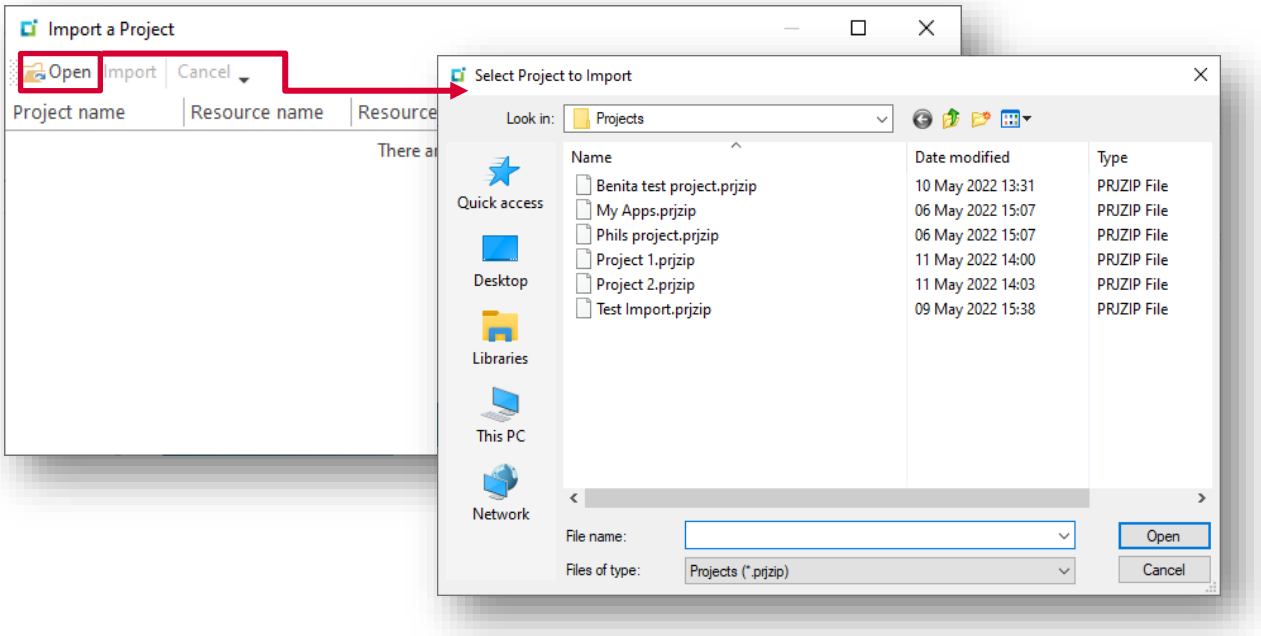
The ZIP file can now be sent (via email, for example) to the end-user.

Importing a Project

An end-user can now import a project by clicking on the **Import Project** button in the Administration tab of the ribbon bar:

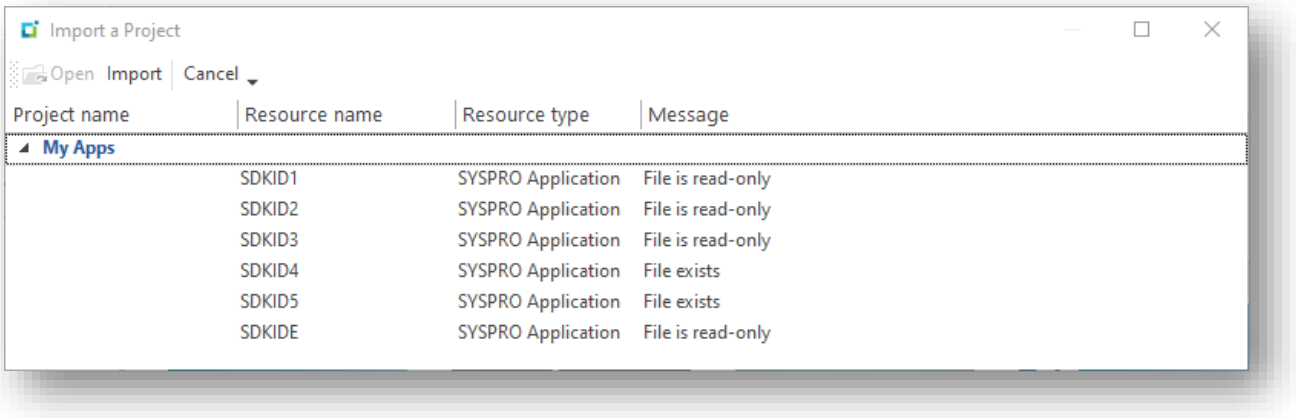


Once the Project Window is displayed, select the **Open** button to locate the ZIP file:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

The contents of the project are shown in the Resource List:

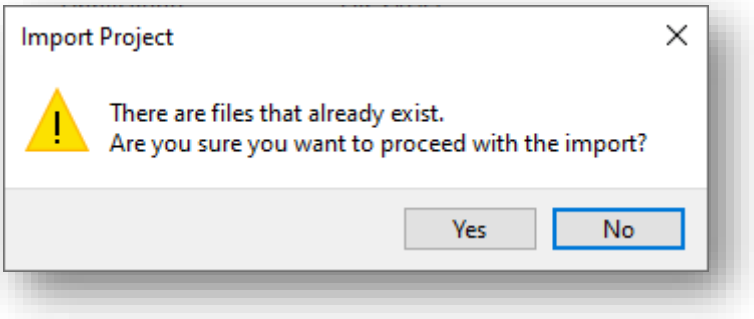


Select **Import** to import the resource contents.

- The files are unzipped from the ZIP file into a temporary folder in the …`\Projects` folder and are then copied to their respective locations on the client machine.
- If the system is running in client/server mode, the files are then copied back to the server machine, again to their respective folders.

Files imported into the …`\AppDesigner\Resources` folder on the server are automatically self-healed to other client machines as users log in (in much the same manner as the contents of `ManagedAssemblies` and `VBSModules` folders).

Note: If any files are read-only, then the import will not be allowed to happen.
If any files already exist, the user will be prompted to decide whether to continue with the import:



USING THE {RESOURCE} KEYWORD

You can define a path using the keyword **`{resources}`** for these various controls:

- Flowgraph
- Notepad
- Image viewer
- Document viewer
- Syntax text editor
- XAML control

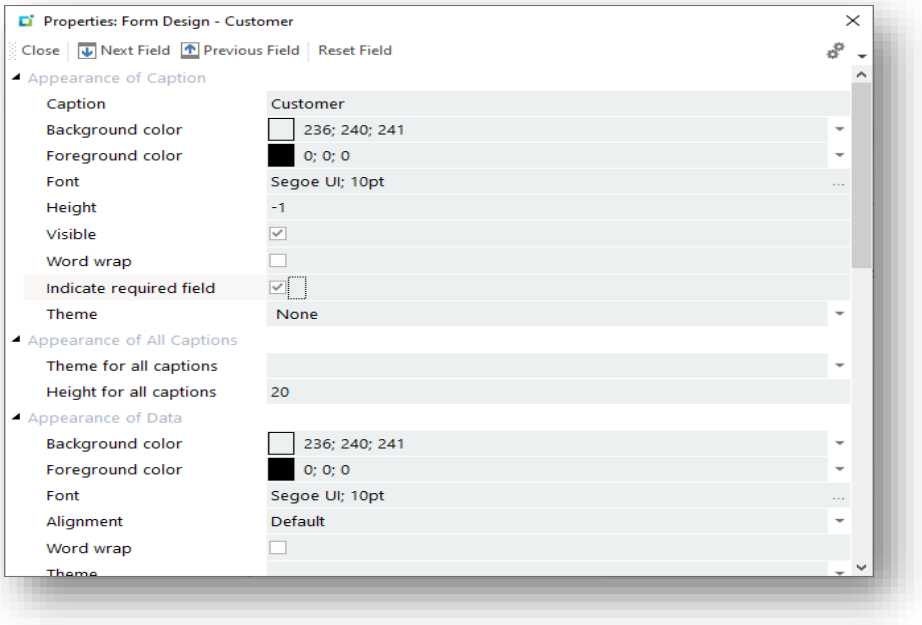
© SYSPRO. All Rights Reserved. All trademarks are recognized.

These controls allow a specific file to be defined that will be loaded when the application runs. However, the entered path may not exist on another computer and therefore the application will fail to load the file.

To overcome this, you can enter a path starting with **`{resources}`**. This keyword will be changed at runtime (and when designing the application) to point to the `…settings\AppDesigner\Resources` folder on both the local and target computer.

You would store your file(s) in this `Resources` folder and, using **Projects**, you can package up your solution including your resource files and send your solution as a ZIP file to the target machine for importing.

Here's an example of entering a path for the Flowgraph control:



© SYSPRO. All Rights Reserved. All trademarks are recognized.

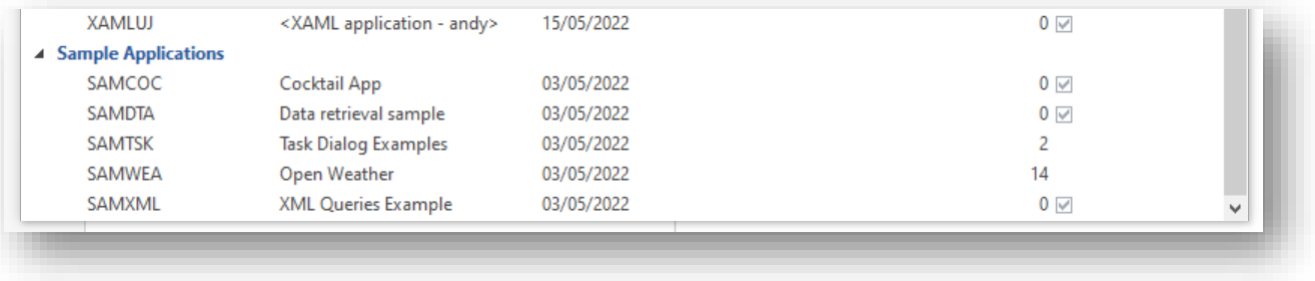
Sample Apps

There are a series of sample Apps that ship with the product.

These Applications are intended to show different features of the designer and its controls and are fully working Applications.

The code is well documented and should be a good source of examples and techniques.

Sample Applications can be viewed when the **Open** button is pressed:



Sample Apps are read only and cannot be amended. However, they can be copied by using the copy function at the top of the **Open Application** pane and renamed; thereafter they can be changed.

© SYSPRO. All Rights Reserved. All trademarks are recognized.

Appendix A – VBScript code snippets

This appendix contains some useful code snippets:

App Comment section

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''
'App Name:
'Date:
'Authored by:
''''''''''''''''''''''''''''''''''''''''''''''''''''''
'This application ……
```

Catching EventType

```
'check for toolbar events
if EventType = ToolbarEvent then
  select case EventID
    case Tbar_Close
        call CallSYSPROFunction(CloseApp, " ", " ")

    case Tbar_aaaaa
        'do something here

    end select

end if
```

Displaying OnUserEvent Parameters

```
'comment out any line you do not want to display
dim msg
msg = ""
msg = msg & "EventType: " & EventType & vbcrlf
msg = msg & "EventName: " & EventName & vbcrlf
msg = msg & "EventID: " & EventID & vbcrlf
msg = msg & "Param1: " & param1 & vbcrlf
msg = msg & "Param2: " & param2 & vbcrlf
msg = msg & "Param3: " & param3 & vbcrlf

msgbox msg
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

Get Local Windows Temp Directory Name

```
''''''''''''''''''''''''''''''''''''''''''''''''
'Get local windows temp directory
''''''''''''''''''''''''''''''''''''''''''''''''
dim fso, tempdir
set fso = CreateObject("Scripting.FileSystemObject")
tempdir = fso.GetSpecialFolder(2)  'gets local temp directory
```

Toolbar Constants

```
'toolbar event ids to help make code more readable
const Tbar_File = "01001"
const Tbar_Close = "01002"
const Tbar_aaaaa = "01003"
```

Writing Text to a File – Simple Function

```
function WriteTextToFile(strtext, strfilename)
dim fso
dim f
set fso = CreateObject("Scripting.FileSystemObject")
set f = fso.CreateTextFile(strfilename,true)
f.WriteLine(strtext)
f.Close

end function
```


© SYSPRO. All Rights Reserved. All trademarks are recognized.

SYSPRO™

www.syspro.com

Copyright © SYSPRO. All rights reserved.
All brand and product names are trademarks or
registered trademarks of their respective holders.