Chapter 6
Adding Controls to Your Windows

Objective

This lesson examines the many controls you can add to your windows. CA-Visual Objects provides several predefined classes that your program can use to create these controls. When you finish this lesson, you should be familiar with all of the controls on the Window Editor Tool Palette.

Overview

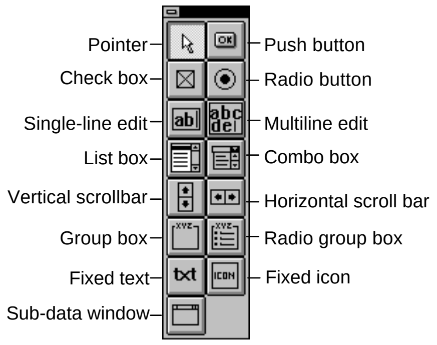
Controls allow users to communicate with an application. They can be placed on both data and dialog windows. The major difference is that on dialog windows, the data associated with a control is buffered by the control, while data window controls can be tied directly to a field in a data server. Also, some controls are tied specifically to a data window (like a subdata window control).

In the previous lesson, you briefly saw single-line edit controls. The following exercise provides greater detail about the controls listed below:

- Single-line edit
- Multiline edit
- Combo box
- Check box
- Radio button
- Radio button group
- List box

- Group box
- Fixed icon
- Push button

All of these controls can be created, using the Window Editor Tool Palette, shown in the figure below:



You may have wondered how the data window in the previous lesson is capable of storing and retrieving data between the database and the controls on the Customer window. Essentially, the data window uses name-based linkages to the data server.

If the control's name has a corresponding field in the data server, the data window automatically associates the two when retrieving and storing data. If a control has no corresponding field in a data server, the data held in the control is not automatically retrieved from, or stored to, a data file, and consequently, must be manipulated by your program.

When you created the Customer window using Auto Layout, single-line edit controls were created based on the Customer DB server. Each of these data controls are named according to the fields in the data server.

Exercise

Single-Line Edit (SLE) Controls

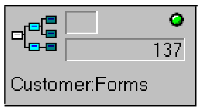
Single-line edit controls are the most widely used type of data-entry control. These controls are ideally suited to fields such as names, descriptions, numeric amounts, and dates.

Data is displayed in the control as text. The user can enter and delete characters, as well as cut, copy, and paste text. The characters in the control are formatted according to the field specification of its corresponding field in the database, or according to an attached FieldSpec object.

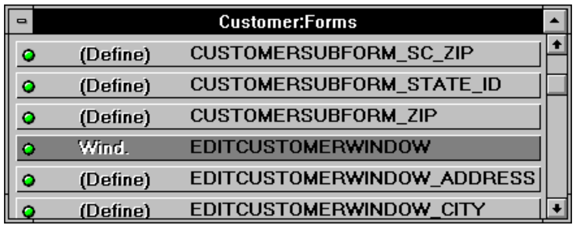
Single-line edit controls can be used to capture text, numeric, date, and logical types of data. Any necessary conversions are done automatically by the data window.

The following exercise demonstrates the use of single-line edit controls as data servers:

1. Open the Customer:Forms module by double -clicking its button.



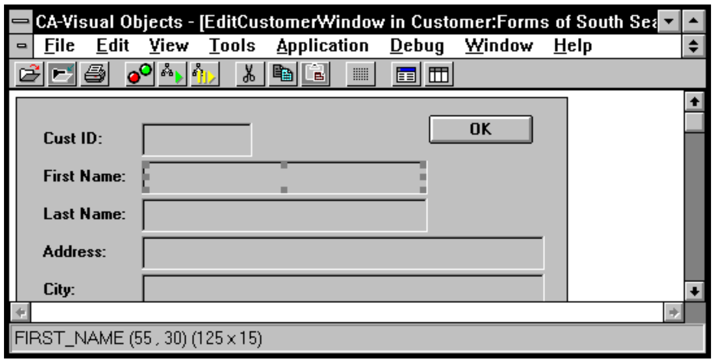
2. Open the EditCustomerWindow window entity by double -clicking on it.



Customer:Forms

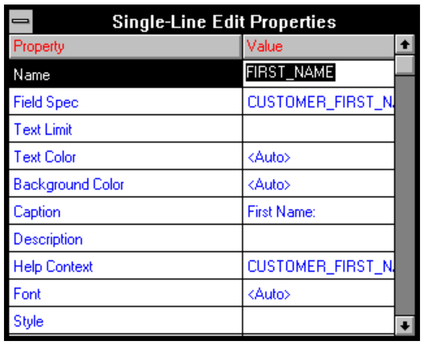
(Define)	CUSTOMERSUBFORM_SC_ZIP
(Define)	CUSTOMERSUBFORM_STATE_ID
(Define)	CUSTOMERSUBFORM_ZIP
Wind.	EDITCUSTOMERWINDOW
(Define)	EDITCUSTOMERWINDOW_ADDRESS
(Define)	EDITCUSTOMERWINDOW_CITY

3. Select the single-line edit control to the right of the First Name: label by clicking on it.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Sea
File Edit View Tools Application Debug Window Help
Cust ID:
OK
First Name:
Last Name:
Address:
City:

FIRST_NAME (55 , 30) (125 x 15)

4. Inspect the Name property in the Single-Line Edit Properties window. This represents the name of the field in the Customer data server (in this case, First_Name).



Single-Line Edit Properties

Property	Value
Name	FIRST_NAME
Field Spec	CUSTOMER_FIRST_N
Text Limit	
Text Color	<Auto>
Background Color	<Auto>
Caption	First Name:
Description	
Help Context	CUSTOMER_FIRST_N
Font	<Auto>
Style	

Multi Line Edit (MLE) Controls

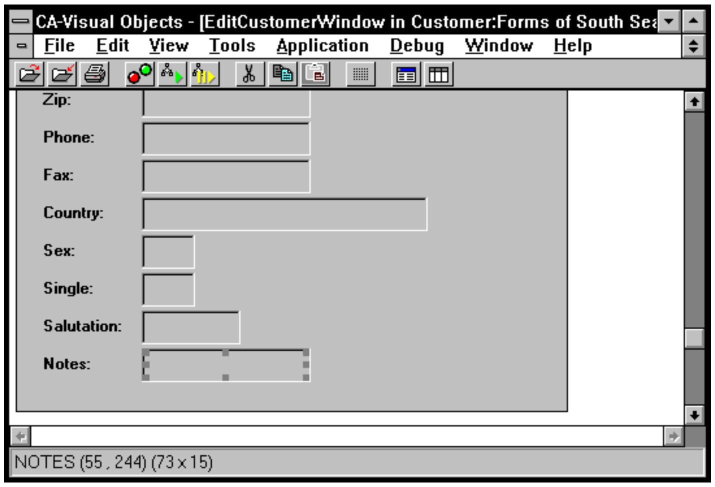
A multiline edit control differs from a single-line edit control in its ability to accept multiple lines of text. Editing functions (like cut and paste) are also available within this control. Multiline edit controls are suited to descriptive information, such as comments, notes, and addresses.

Creating the MLE Control

Now, let's change the Notes single-line edit control on the EditCustomerWindow to a multiline edit control. The Notes field is a memo field, and therefore lends itself well to a multiline edit control.

1. To delete the Notes single-line edit control and the Note: label, click on each and press the Delete key.

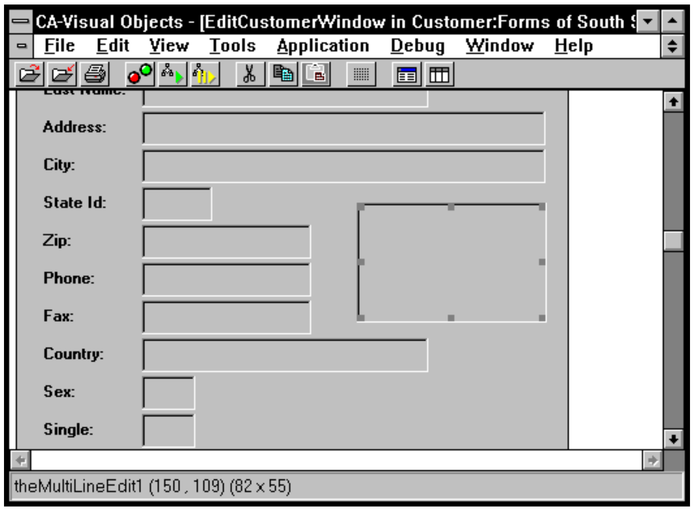
   Note: If you hold the Ctrl key while clicking on these controls, both are selected and may be deleted simultaneously.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Sea
File Edit View Tools Application Debug Window Help
Zip:
Phone:
Fax:
Country:
Sex:
Single:
Salutation:
Notes:
NOTES (55, 244) (73 x 15)

2. Click on the multiline edit control Tool Palette button.

   This selects a multiline edit control.

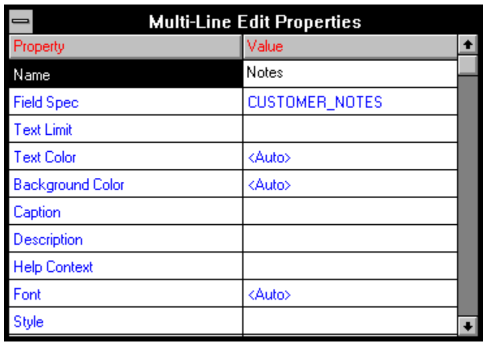
3. Place the control by clicking on the window canvas area. Position it to the right of the *State* single-line edit control, about halfway across the window canvas area. Size it so that it is about 1 inch high and 2 inches wide.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South
File Edit View Tools Application Debug Window Help
Address:
City:
State Id:
Zip:
Phone:
Fax:
Country:
Sex:
Single:

theMultiLineEdit1 (150 , 109) (82 x 55)

This control has not been tied to a field in a table yet. In the Properties window, notice it is named theMultiLineEdit1.

4. Change the Name property to **Notes**. This corresponds to the Notes field in the Customer table.


Multi-Line Edit Properties

Property	Value
Name	Notes
Field Spec	CUSTOMER_NOTES
Text Limit	
Text Color	<Auto>
Background Color	<Auto>
Caption	
Description	
Help Context	
Font	<Auto>
Style	

5. You also want it to inherit specific field spec properties. In the Properties window, change the FieldSpec property to **CUSTOMER_NOTES**.

Viewing Your Results

You can now build and run the application to ensure that your changes worked. Do not close the Window Editor—since you will return here for the following exercises.

1. Select the Build button from the toolbar. (This also performs a "save" operation.)
2. Select the Execute button from the toolbar.
3. When the application starts, select the Open command from the Files menu and open the Customer file.
4. Click the Edit toolbar button and you will see the window you are working on.

   The multiline edit control should now be visible in the window.
5. When you are finished, return to the EditCustomerWindow Window Editor session for the next set of changes.

Combo Box Controls

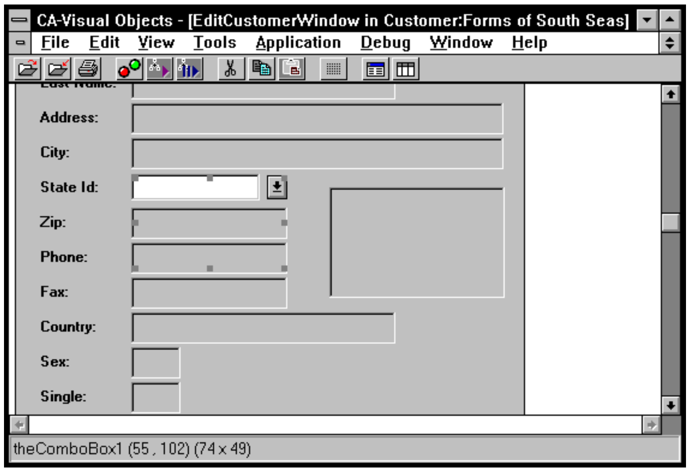
Using combo box controls, you can avoid the need for a user to know the exact data your program expects, as well as having to write validation code. For example, if the State_ID control were to remain a single-line edit control, it would require the user to be familiar with the abbreviations for 50 U.S. states. Additionally, every entry would have to be validated to insure that a valid code was entered.

We will now create a combo box control for the State_ID field:

1. Delete the State_ID single-line edit control by clicking on it and pressing the Delete key.
2. Click on the combo box Tool Palette button.

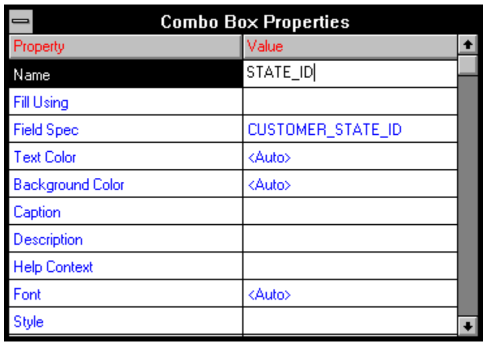
   This selects a combo box control.

3. Drop the control by clicking on the window canvas area. Position it next to the State ID: label. Size it roughly as shown in the figure below:


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File Edit View Tools Application Debug Window Help
Address:
City:
State Id:
Zip:
Phone:
Fax:
Country:
Sex:
Single:
theComboBox1 (55 , 102) (74 x 49)

This control has not been tied to a field in any table yet. In the Properties window, notice it is named theComboBox1.

4. Change the Name property to **STATE_ID.** This ensures that the data window stores and retrieves values to and from the State_ID field of the Customer data server.

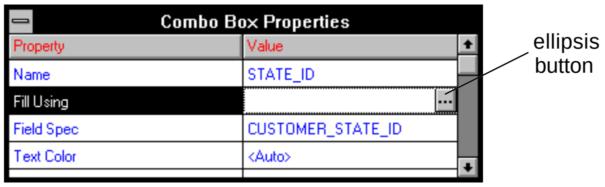


Property	Value
Name	STATE_ID
Fill Using	
Field Spec	CUSTOMER_STATE_ID
Text Color	<Auto>
Background Color	<Auto>
Caption	
Description	
Help Context	
Font	<Auto>
Style	

5. You also want it to inherit specific field spec properties. In the Properties window, change the FieldSpec property by selecting CUSTOMER_STATE_ID from the drop-down list box.

This ensures that any special validations applied to the field specification are tied to your combo box.

6. Now, you need to tell the combo box what to display. Select the Fill Using property, then click on the ellipsis button.

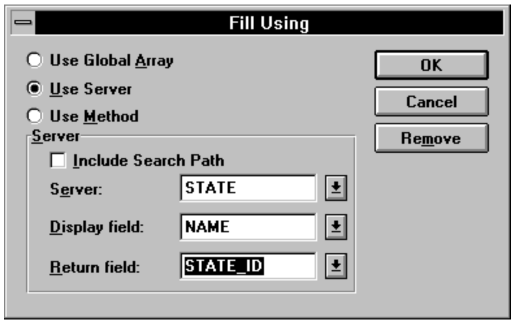


Property	Value
Name	STATE_ID
Fill Using	
Field Spec	CUSTOMER_STATE_ID
Text Color	<Auto>

You are prompted by the Fill Using dialog box. The Fill Using property of the combo box allows you to fill its list with either an array, the contents of a data server, or a method.

The State table and server have already been created and populated with valid state codes and names, so let's use it here.

7. Select the Use Server radio button.



○ Use Global Array
● Use Server
○ Use Method

Server
☐ Include Search Path
Server: STATE
Display field: NAME
Return field: STATE_ID

8. From the Server combo box, select STATE.

The Server group box also provides the option to display one server field and return another to the field attached to the control. This feature allows you to retain your normalized databases while showing the user more descriptive information. In this case, let's display the state name instead of the state code.

9. Select NAME from the Display field combo box.

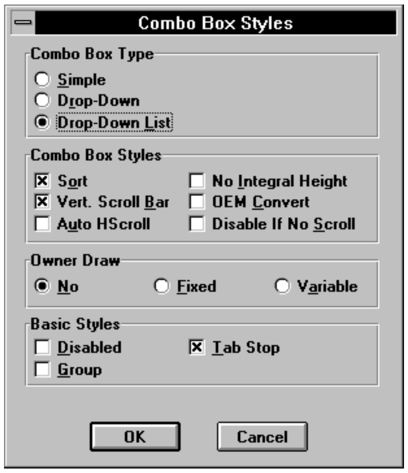
10. Select STATE_ID in the Return field combo box, since it is the only field that appears in both servers.

11. Choose OK to close the Fill Using dialog box.

To disable a user's ability to enter invalid codes, you can change the style on the combo box to Drop-Down List. Using this type of combo box forces the user to select a value in the list.

1. Double-click on the State_ID combo box on the window canvas area.

   The Combo Box Styles dialog box appears:

   Combo Box Styles

   Combo Box Type: Simple [✓ Drop-Down List] Drop-Down

   Combo Box Styles: [✓ Sort] [✓ Vert. Scroll Bar] Auto HScroll; No Integral Height; OEM Convert; Disable If No Scroll

   Owner Draw: [✓ No] Fixed Variable

   Basic Styles: Disabled; Group; [✓ Tab Stop]

2. Select the Drop-Down List radio button in the Combo Box Type group box.

   You may also want to ensure that your list items get sorted alphabetically when the list is displayed.

3. Select the Sort check box in the Combo Box Styles group box.

4. Choose OK.

Note: You can now build and run the application as before to test your changes. When you are finished, return to the EditCustomerWindow Window Editor session for the next set of changes.

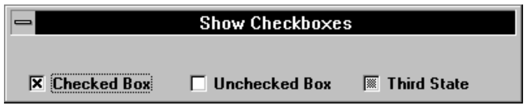
Check Box Controls

A check box is a square box with associated text that usually appears to the right of the check box. It acts as a toggle switch, allowing a user to turn an option on or off. Thus, it is usually used to represent logical fields.

When the check box is linked to a logical field in the data server, a value of TRUE in the server represents the checked (or on) state of the check box, while a value of FALSE represents the unchecked (or off) state.

One of the properties of a check box provides for a three-state check box. The third state is dimmed and indicates that the check box status is unknown (or undefined).

The check box states are shown in the following dialog box:



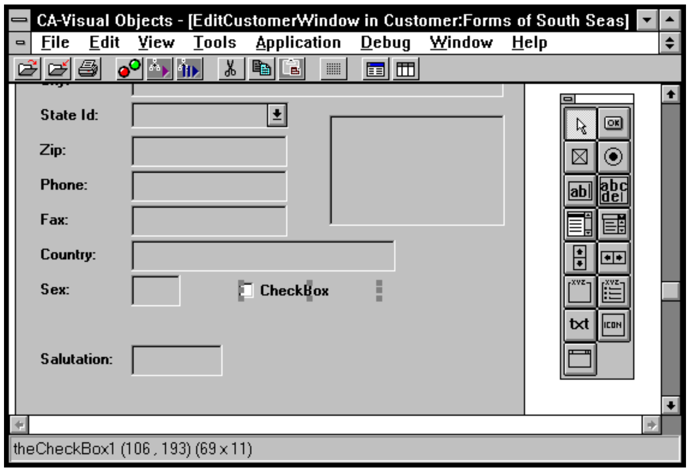
The Customer data server has a logical field named "Single" to represent marital status. Now, we will put a check box control beside the Sex field to represent single status:

Note: More recent releases of CA-Visual Objects 1.0 may already have automatically placed a check box control for this field on the window. In this case, for the purposes of this exercise, delete the check box and then continue with Step 2.

1. Delete the Single: single-line edit control and label by clicking on them and pressing the Delete key.
2. Click on the check box Tool Palette button.

   This selects a check box control.

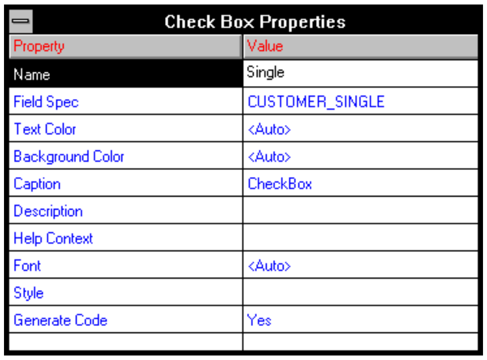
3. Place the control by clicking on the window canvas area. Position it next to the Sex: label.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File Edit View Tools Application Debug Window Help
State Id:
Zip:
Phone:
Fax:
Country:
Sex:
CheckBox
Salutation:

theCheckBox1 (106 , 193) (69 x 11)

This control has not been tied to a field in a table yet. In the Properties window, notice it is named theCheckBox1.

4. Change the Name property to **Single**.


Check Box Properties

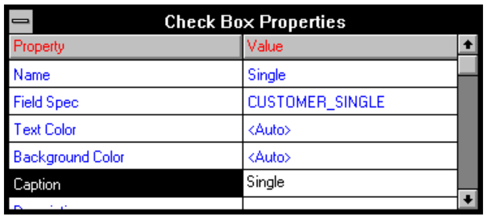
Property	Value
Name	Single
Field Spec	CUSTOMER_SINGLE
Text Color	<Auto>
Background Color	<Auto>
Caption	CheckBox
Description	
Help Context	
Font	<Auto>
Style	
Generate Code	Yes

This ensures that the data window stores and retrieves values to and from the Single field of the Customer data server.

5. We also want it to inherit specific field spec properties. In the Properties window, change the FieldSpec property to **CUSTOMER_SINGLE**.

   This ensures that any special validations applied to the field specification are always tied to your check box.

6. We also want to change the caption to reflect the true nature of the field. In the Properties window, change the Caption property to **Single**.



Check Box Properties

Property	Value
Name	Single
Field Spec	CUSTOMER_SINGLE
Text Color	<Auto>
Background Color	<Auto>
Caption	Single

Radio Button and Radio Button Group Controls

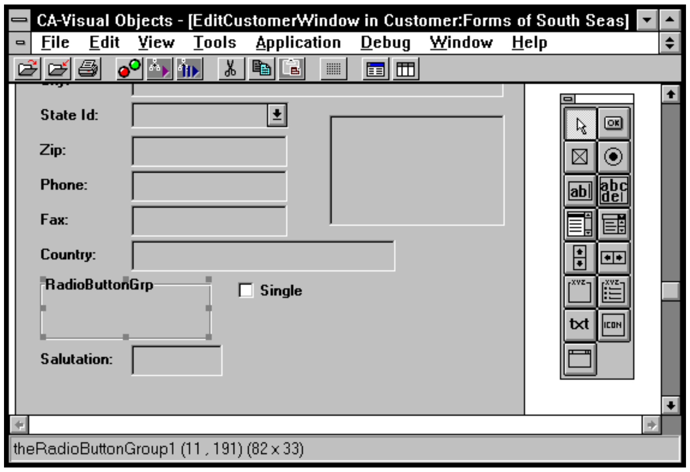
In radio button group controls, individual radio buttons provide mutually exclusive responses to a condition where only one choice is required. When you click a radio button, it is checked (or on). If you then click another radio button within the same radio button group, the radio button you first clicked on is unchecked (or off).

The radio button is an oddity in the Button class, since it is not tied directly to a data field. Instead, it is grouped (with other radio buttons) into a radio button group control. Unless you create distinct radio button groups, all the radio buttons on a window are members of the same radio button group. A radio button group control can be tied to an actual field in a data server.

In the Window Editor, each radio button can be assigned a group value. When retrieving data from the server, the radio button group selects the radio button whose group value corresponds to the data value in the server. When storing data to the server, the radio button group uses the group value of the currently selected radio button.

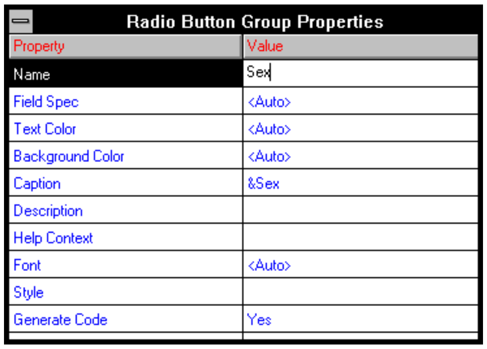
You will now attach a radio button group, with Male and Female options, to the Sex field on the Customer window. The Sex field is an ideal candidate for radio buttons, since there are only two choices, Male and Female.

1. Select and delete both the Sex: label and single-line edit control.
2. Select the radio button group Tool Palette button and then move the mouse to the desired location on the window canvas area. Click the mouse button again to place and position the radio button group box.
3. Size the radio button group so that it is similar to the following:



theRadioButtonGroup1 (11 , 191) (82 x 33)

4. The radio button group is the control that gets linked to the data server. In the Properties window, change the Name property to **Sex**.



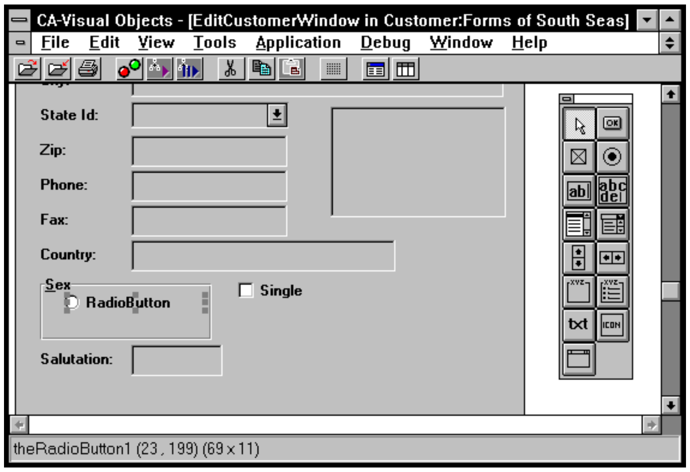
Radio Button Group Properties

Property	Value
Name	Sex
Field Spec	<Auto>
Text Color	<Auto>
Background Color	<Auto>
Caption	&Sex
Description	
Help Context	
Font	<Auto>
Style	
Generate Code	Yes

5. Change the Caption property to **&Sex**.

   Notice that the S is now underlined. This allows the user to press Alt+S to move directly to the control. Now, let's place two radio buttons inside of the radio button group box. The radio buttons are able to be placed one above the other in a column.

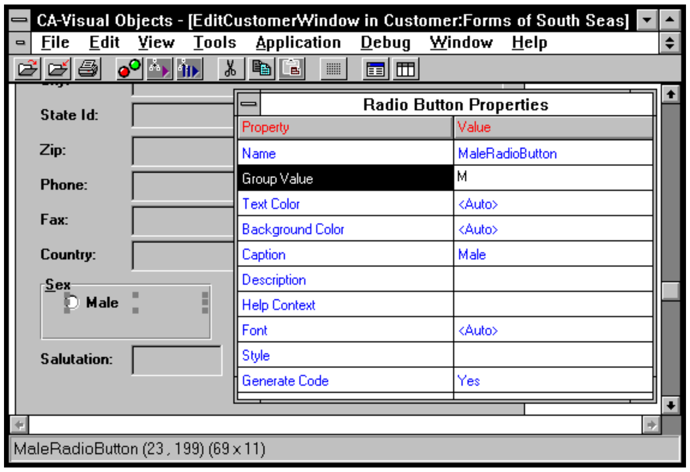
6. Click the radio button Tool Palette button and place a radio button in the radio button group box as shown below:


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File Edit View Tools Application Debug Window Help
State Id:
Zip:
Phone:
Fax:
Country:
Sex
RadioButton
Single
Salutation:

theRadioButton1 (23 , 199) (69 x 11)

This radio button will be used for the Male option. The control is not directly linked to the data server, therefore its name should differ from the fields of the database. Although, it is not necessary to rename the field, it is still a good idea to do so.

7. Type **MaleRadioButton** in the Name property cell.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File Edit View Tools Application Debug Window Help
State Id:
Zip:
Phone:
Fax:
Country:
Sex
Male
Salutation:
Radio Button Properties

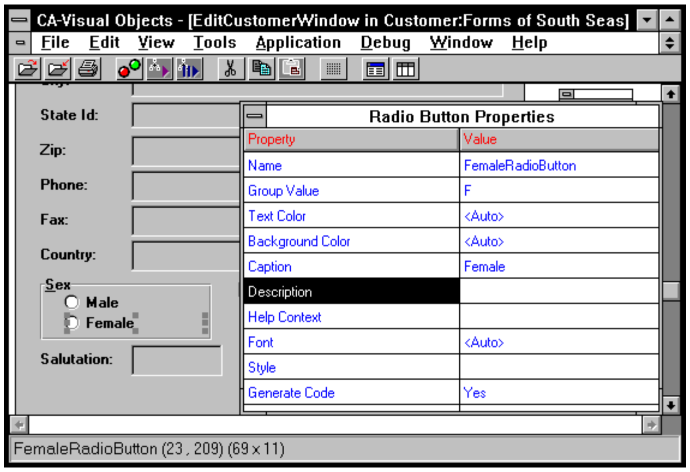
Property	Value
Name	MaleRadioButton
Group Value	M
Text Color	<Auto>
Background Color	<Auto>
Caption	Male
Description	
Help Context	
Font	<Auto>
Style	
Generate Code	Yes

MaleRadioButton (23 , 199) (69 x 11)

8. The Group Value property contains the actual value that is used by the radio button group. Click on this property and type **M**.

9. Now, click on the Caption property and type **Male**. This is displayed on the window, to the right of the radio button it represents.

10. Add the Female radio button following steps 7 through 9. Place it below the Male radio button. You can name the control **FemaleRadioButton**. Set the Group Value property to **F** and the Caption to **Female**.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File Edit View Tools Application Debug Window Help
State Id:
Zip:
Phone:
Fax:
Country:
Sex
Male
Female
Salutation:

Radio Button Properties

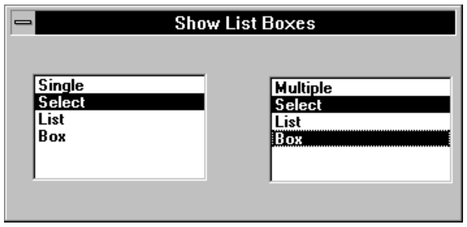
Property	Value
Name	FemaleRadioButton
Group Value	F
Text Color	<Auto>
Background Color	<Auto>
Caption	Female
Description	
Help Context	
Font	<Auto>
Style	
Generate Code	Yes

FemaleRadioButton (23 , 209) (69 x 11)

Note: You can now build and run the application as before to test your changes. When you are finished, return to the Window Editor for the next set of changes.

List Box Controls

The list box control is a collection of text strings. It is displayed as a scrollable, columnar list within a rectangle. A list box can allow either a single selection or multiple selections.



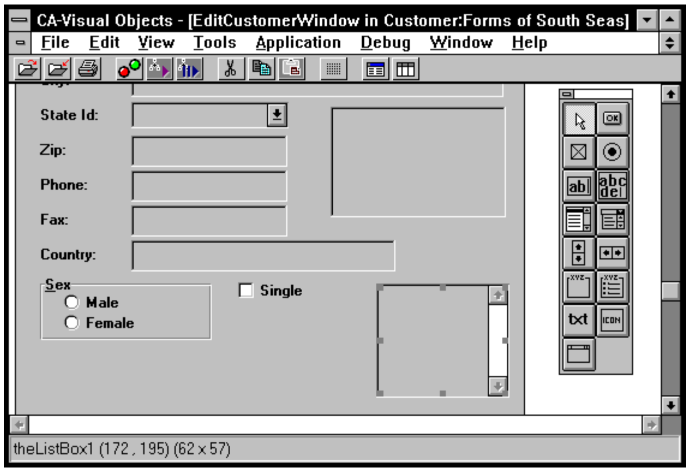
In a single selection list box, the user can select the item that the cursor is on by pressing the spacebar or clicking the left mouse button. In a multiple selection list box, the spacebar or mouse button toggles the selected state.

Navigation is accomplished by use of the vertical scroll bar and the navigation keys (Up/Down arrow and PageUp/PageDown keys) if there are more elements than can fit in the display area. Pressing a letter key moves the cursor and the selection highlight bar to the first item in the list starting with that letter.

You will use a list box to represent the salutation field of the customer data server. The list box has to represent the Mr., Mrs., and Ms. salutations.

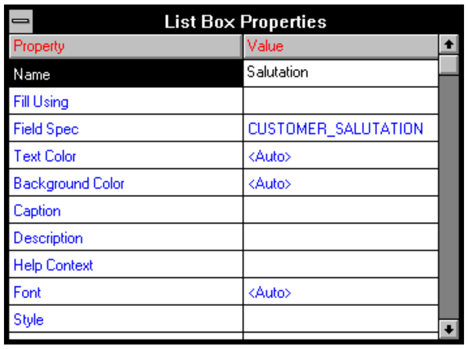
Note: You could have used a combo box or a radio button group to represent this just as easily. Because you have already seen those controls, let's use the list box control here:

1. Delete the Salutation single-line edit control and label by clicking on each and pressing the Delete key.
2. Select the list box Tool Palette button. This selects a list box control.
3. Place the control by clicking on the window canvas area. Position it to the right of the Country field as shown in the following figure. Size it to about 1.5 inches high by 1 inch.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File
Edit
View
Tools
Application
Debug
Window
Help
State Id:
Zip:
Phone:
Fax:
Country:
Sex
Male
Female
Single
theListBox1 (172, 195) (62 x 57)

This control has not yet been linked to a field in a table. In the Properties window, notice it is named theListBox1.

4. Change the Name property to **Salutation**.



List Box Properties

Property	Value
Name	Salutation
Fill Using	
Field Spec	CUSTOMER_SALUTATION
Text Color	<Auto>
Background Color	<Auto>
Caption	
Description	
Help Context	
Font	<Auto>
Style	

This ensures that the data window stores and retrieves values to and from the Salutation field of the Customer data server.

5. You also want it to inherit specific field spec properties. In the Properties window, change the FieldSpec property to **CUSTOMER_SALUTATION**.

   This ensures that any special validations applied to the field specification are always tied to your list box.

   Now, we must create an array for use in filling the list box.

6. Select Modules of South Seas from the Window menu to return to the Module Browser.

7. Select the App:Start() module and click on the Open All Source toolbar button.

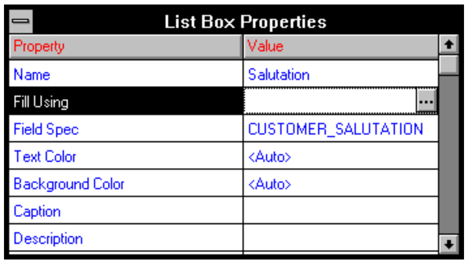
8. Press the Enter key to create a blank line. On this blank line, type the following line of code:

```
GLOBAL GlobalArraySalutation := {"Mr.","Mrs.",;
    "Ms."} AS ARRAY
```

9. Close the Source Code Editor by double-clicking on its system menu. Choose Yes to save the revised source code.

10. Return to the Window Editor by selecting its session from the Window menu.

11. Now we need to tell the list box what to display. Select the Fill Using property and then click on the ellipsis button.

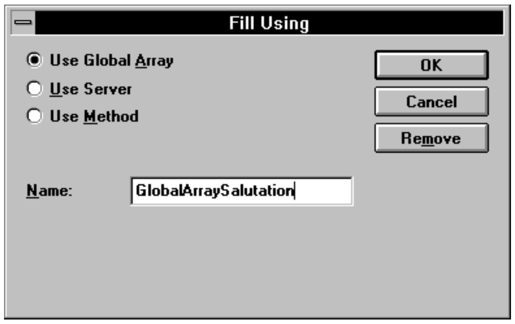


List Box Properties

Property	Value
Name	Salutation
Fill Using	...
Field Spec	CUSTOMER_SALUTATION
Text Color	<Auto>
Background Color	<Auto>
Caption	
Description	

You are prompted with the Fill Using dialog window. The Fill Using property allows you to fill the list box list with either an array, the contents of a data server, or from a specified method.

12. Select the Use Global Array radio button and type **GlobalArraySalutation** in the Name edit control. Then choose OK.



Fill Using

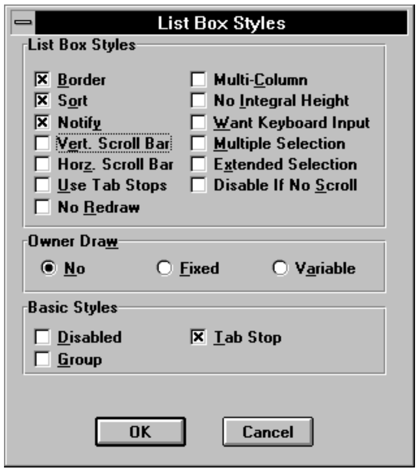
(●) Use Global Array
( ) Use Server
( ) Use Method

Name: GlobalArraySalutation

Let’s look at the available styles for the list box.

13. Double-click the Salutation list box control on the window canvas area.

    The List Box Styles dialog box appears:



List Box Styles

List Box Styles

☒ Border ☐ Multi-Column
☒ Sort ☐ No Integral Height
☒ Notify ☐ Want Keyboard Input
☐ Vert. Scroll Bar ☐ Multiple Selection
☐ Horz. Scroll Bar ☐ Extended Selection
☐ Use Tab Stops ☐ Disable If No Scroll
☐ No Redraw

Owner Draw

◉ No ○ Fixed ○ Variable

Basic Styles

☐ Disabled ☒ Tab Stop
☐ Group

14. Deselect the Vert. Scroll Bar check box in the List box Styles group. You do not need the scroll bar because all of the items fit into the display area.

    You also want to ensure that your list items get sorted alphabetically when the list is displayed.

15. Select the Sort check box in the List Box Styles group and then choose OK.

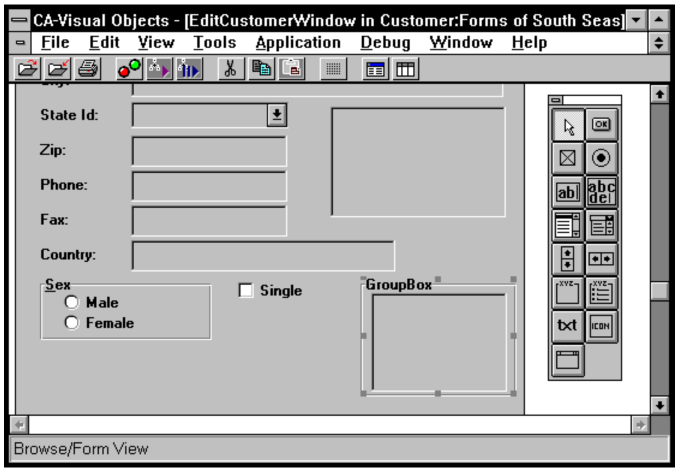
Note: You can now build and run the application as before to test your changes. When you are finished, return to the EditCustomerWindow window canvas area for the next set of changes.

Group Box Controls

The group box control has no relation to data fields or variables. It is used only to visually group controls, however, it can affect the controls that are within it by means of its tab and group style settings.

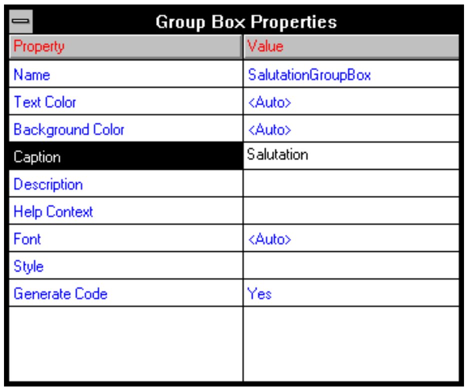
The group box allows you to add a labeled box to a window. These are useful for enhancing the aesthetic quality of a window, for setting up tab stops, and for making certain groups of controls unselectable.

1. Select the group box Tool Palette button and place a group box control on top of the Salutation list box.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File Edit View Tools Application Debug Window Help
State Id:
Zip:
Phone:
Fax:
Country:
Sex
Male
Female
Single
GroupBox
Browse/Form View

2. Size the group box control so that it surrounds the list box.

3. Change Name property to **SalutationGroupBox** and the group box Caption property to **Salutation**.



Group Box Properties

Property	Value
Name	SalutationGroupBox
Text Color	<Auto>
Background Color	<Auto>
Caption	Salutation
Description	
Help Context	
Font	<Auto>
Style	
Generate Code	Yes

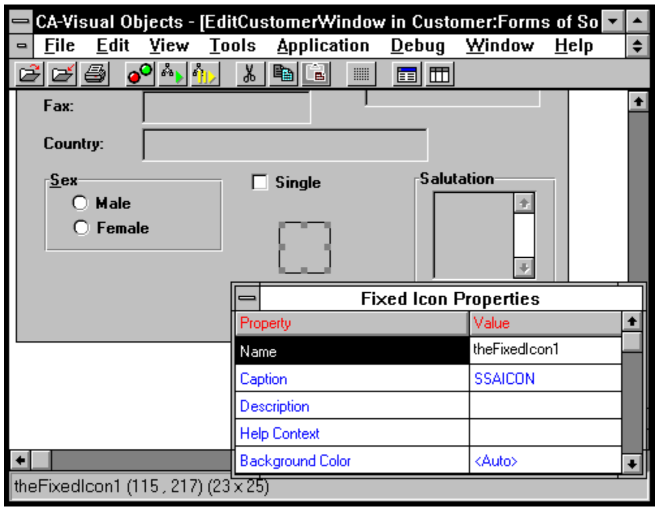
Note: You can now build and run the application as before to test your changes. When you are finished, return to the EditCustomerWindow window canvas area for the next set of changes.

Fixed Icon Controls

The next control we will look at is the fixed icon control. This control allows you to add icons to your data windows and dialog windows. The purpose of the fixed icon control is for aesthetic purposes only.

1. Click the Fixed Icon Tool Palette button to select it, and then move the mouse to the desired location on the window canvas area (any free spot on the window is fine).
2. Click the mouse button again.

This places a fixed icon control on the window canvas area:


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of So
File Edit View Tools Application Debug Window Help
Fax:
Country:
Sex
Male
Female
Single
Salutation
Fixed Icon Properties

Property	Value
Name	theFixedIcon1
Caption	SSAICON
Description	
Help Context	
Background Color	<Auto>

theFixedIcon1 (115 , 217) (23 x 25)

3. In the Fixed Icon Properties window, change the Caption property to **SSAICON**.

   The Caption property holds the name of an icon that is already a part of, or associated with, your application.

Note: You can now build and run the application as before to test your changes. When you are finished, return to the EditCustomerWindow window canvas area for the next set of changes.

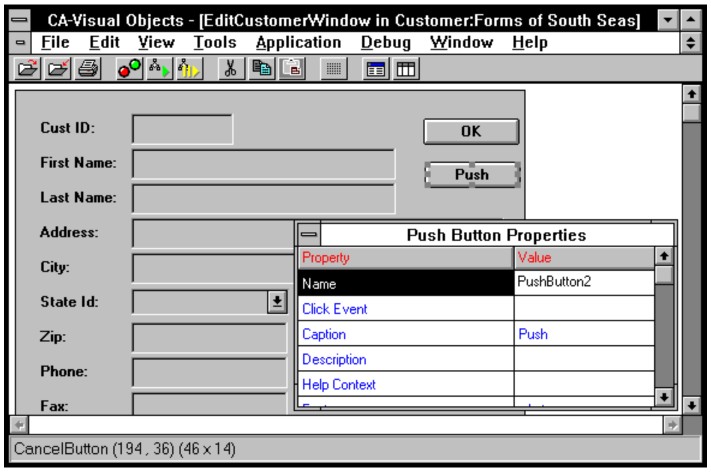
Push Button Controls

Push button controls are command controls that trigger an action without retaining any type of on/off indication. There are two types of push buttons. Standard option push buttons are the most commonly used. Default option push buttons have a slightly thicker border, and may be activated by the Enter key whenever a non-push button control has input focus. As this implies, only one push button on a dialog window should have the default option type.

Generally, a dialog window has an OK button and a Cancel button to accept or abort whatever the dialog window is trying to do. The OK button was added in the “Creating and Using Windows” chapter. Now, to add the Cancel push button to your Customer window:

1. Select the Push Button icon on the Tool Palette and then move the mouse to the top-right corner on the window canvas area.

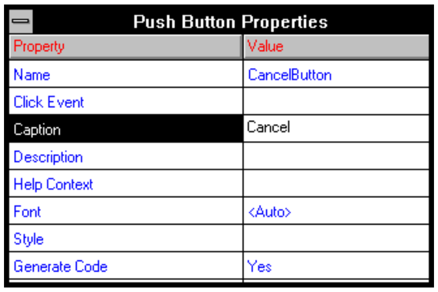
   This places a push button onto your window canvas area, just below the OK button.



Property	Value
Name	PushButton2
Click Event	
Caption	Push
Description	
Help Context	

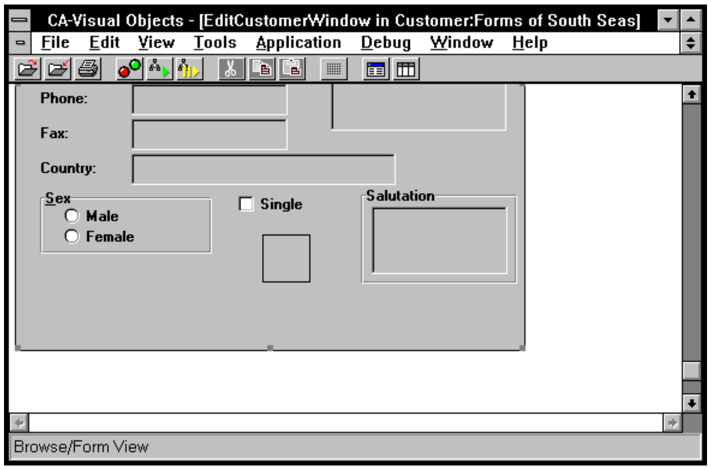
CancelButton (194, 36) (46 x 14)

2. In the Push Button Properties dialog box, change the Name property to **CancelButton**. Also, change the Caption property to **Cancel**.

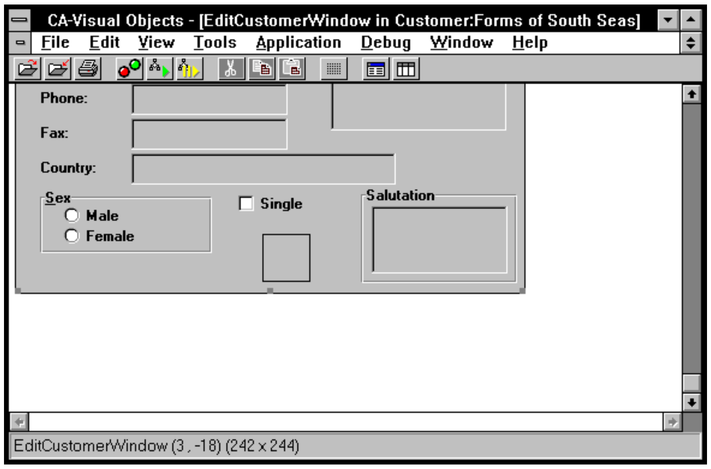

Push Button Properties

Property	Value
Name	CancelButton
Click Event	
Caption	Cancel
Description	
Help Context	
Font	<Auto>
Style	
Generate Code	Yes

3. Scroll down so you can see the bottom of the window.


CA-Visual Objects - [EditCustomerWindow in Customer:Forms of South Seas]
File Edit View Tools Application Debug Window Help
Phone:
Fax:
Country:
Sex
Male
Female
Single
Salutation
Browse/Form View

4. Select the window by clicking in an area with no controls, and drag the lower border up. Adjust the list box and group box size so that the window looks as follows:



EditCustomerWindow (3 , -18) (242 x 244)

Note: You can now build and run the application as before to test your changes. When you are finished, return to the EditCustomerWindow window canvas area for the next set of changes.

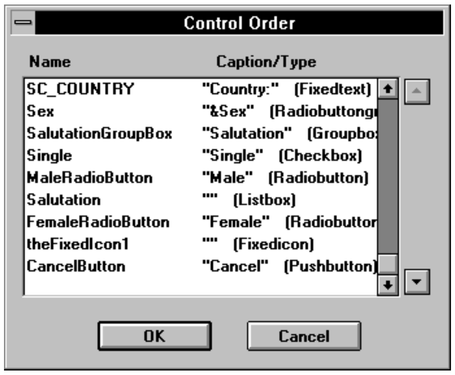
Programming Techniques

The following section deals with some of the more intricate aspects of creating controls for your windows.

Tab and Group Stops

Now that you have created a data entry window, you need to control the order in which a user tabs through the controls. Of course, the Tab key moves you forward through the controls, while Shift +Tab moves back. The default tab order moves top-to-bottom, left-to-right, according to window placement. This may not always be acceptable. For example, if you had a window design consisting of two columns, you may want to move down to the bottom of the left column before moving to the top of the right.

By selecting the Control Order command from the Edit menu, you can see the order in which the controls get focus. If you scroll down to the bottom of the list, you will see the problem.

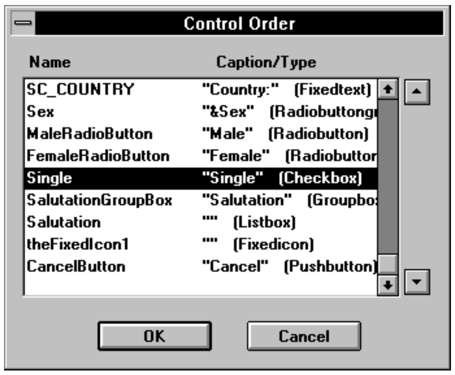


Name	Caption/Type
SC_COUNTRY	"Country:" (Fixedtext)
Sex	"&Sex" (Radiobuttongr
SalutationGroupBox	"Salutation" (Groupbo:
Single	"Single" (Checkbox)
MaleRadioButton	"Male" (Radiobutton)
Salutation	"" (Listbox)
FemaleRadioButton	"Female" (Radiobuttor
theFixedIcon1	"" (Fixedicon)
CancelButton	"Cancel" (Pushbutton)

The MaleRadioButton and the FemaleRadioButton should immediately follow the Sex radio button group control. These should be promoted to reflect this. To do this, perform the following steps:

1. Click on the MaleRadioButton entry and then use the up arrow button (to the right of the vertical scroll bar) to move it just below the Sex entry.
2. Click on the FemaleRadioButton entry and position it just below the MaleRadioButton entry in the list.
3. Click on the Single check box entry and move it up so it follows FemaleRadioButton.

   Your window should now look as follows:



Name	Caption/Type
SC_COUNTRY	"Country:" (Fixedtext)
Sex	"&Sex" (Radiobuttongr
MaleRadioButton	"Male" (Radiobutton)
FemaleRadioButton	"Female" (Radiobutton
Single	"Single" (Checkbox)
SalutationGroupBox	"Salutation" (Groupbo:
Salutation	"" (Listbox)
theFixedIcon1	"" (Fixedicon)
CancelButton	"Cancel" (Pushbutton)

4. Choose OK to close the Control Order dialog box.
5. Close the Window Editor by double-clicking on its system menu and select Yes to save the revised design.

You can force tabbing to occur in any order that you wish, based on the order you specify. But how do you decide what gets focus and what does not? The operating system provides all the logic to move input focus from one control to another, with a little help from you and from the styles that can be applied to a control.

Double-clicking on any control on your window brings up the Styles dialog box for that control. Although the dialog box for each style has certain differences depending on the control, they all have a Basic Styles section. For example, on the Radio Buttons Styles dialog box, you see three check boxes, corresponding to the following options:

- Disabled

  If checked, this option dims the control to prevent it from obtaining input focus. This is often adjusted programatically.

- Tab Stop

  If checked, this option allows the control to be part of the tabbing sequence.

- Group Stop

  If checked, this option turns the control into a group control. Once a group box has input focus, any command controls inside it are navigable through the cursor keys. This is especially useful in grouping radio buttons.

When you add a control to the window canvas area, the default styles are as follows:

Control Type	**Tab Stop**	**Group Stop**
Push button	Yes	No
Check box	Yes	No
Radio button	Yes	No
Single-line edit	Yes	No
Multiline edit	Yes	No
List box	Yes	No
Combo box	Yes	No
Scroll bars	No	No

Continued

Continued

Control Type	Tab Stop	Group Stop
Radio button group	No	Yes
Group box	No	Yes
Fixed text	No	No
Fixed icon	No	No
Sub data window	N/A	N/A

This works fine for the most part, but here are a few tips:

- You should disable the tab stop on a radio button if it is in a radio button group. Any radio button that is inside that group has an implied tab stop.
- Do not put tab stops on controls that do not allow data entry like fixed text, fixed icon, and group box controls.
- Group stops should be reserved for group box controls and radio button group controls. By the same token, the controls contained in the group should not have a tab stop.

Control Order and Multiple Groups

Control order for multiple groups can also require special attention. The problem is that one group does not end until the next group begins. In short, imagine you have several radio buttons in a radio button group box. The radio button group has the group style selected.

Now let's say that you have a separate set of radio buttons. The default Windows action is to assume that this set of radio buttons belongs to the first radio button group. This is wrong, and to fix it, you would create a second radio button group for the second set of radio buttons. This would cause a new group to start, ending the first group—since every radio button group has the group style checked. If you do not use either a group box or a radio button group, you can set the group style for any other control so it starts a different group.

Naming Controls

Your data-entry window is complete, but before completing this chapter, let’s discuss the naming convention that was used in naming the controls.

For data windows you create that need to update fields in a table, it is easy. You use the field name as the control name, since a name-based association exists with the attached data server to apply changes to fields directly for you.

What about controls that are not linked to fields? They can be called anything, except the name of a field in an attached server. Long names are supported, giving you the ability to create descriptive names. The convention that is used is to name the button according to its type, followed by a string that described the type of control in question. For example:

- A push button labeled Edit is called EditButton
- A radio button labeled Payment is called PaymentRadioButton
- A single-line edit that represented a search string is called SearchSLE
- A list box representing the state is called StateListBox

Using this convention, you see that your control name tells you what the entity represents, as well as the properties you can expect it to have.

Note: You can now build and run the application as before to test your changes. This lesson is complete, so you can close the Window Editor and return to the Module Browser.

Summary

At this point, you have seen most of the window controls in use. You have explored how these controls are interrelated through the use of tab stops and group stops. With these tools at your disposal, you can now generate almost any kind of dialog window for your users.

In the next lesson, you will learn about inheritance and subclassing principles, which are the most important concepts in object-oriented programming.