Chapter 8
Creating Menus and Toolbars

Objective

This lesson introduces you to the basic concepts of menus and toolbars, and how they interact with your application. By the end of this lesson you will:

- Know how to create and customize menus and toolbars
- Understand how menu events invoke methods within your application
- Be able to attach menus and toolbars to your windows
- Know how to customize the behavior of menus

Overview

Menus and toolbars are essential components of a Windows application. Menus allow the user to navigate through the system and issue commands, while toolbars provide quick and easy access to frequently used menu commands within an application.

Each application can have its own menus and toolbars, which are attached to windows. CA-Visual Objects allows you to generate and modify menus and toolbars using the Menu Editor.

Exercise

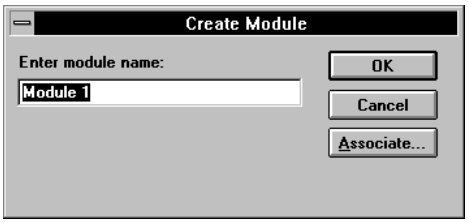
Creating a New Module

Let's begin this exercise by creating a new module, for which we can then create a new menu:

1. Open the South Seas Adventures application by double-clicking on its button in the Application Browser.

2. Create a new module by selecting the New Module toolbar button.

   The Create Module dialog box appears:



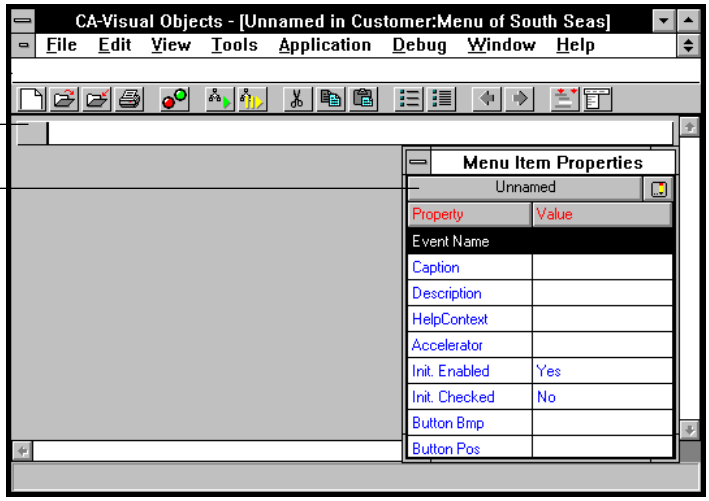
3. Type **Customer:Menu** in the Enter Module Name edit control and choose OK.

   The new Customer:Menu module button appears in the South Seas Adventures Module Browser.

Creating the Menu

1. Select the Open Entity toolbar button and choose Menu Editor from the local pop-up menu.

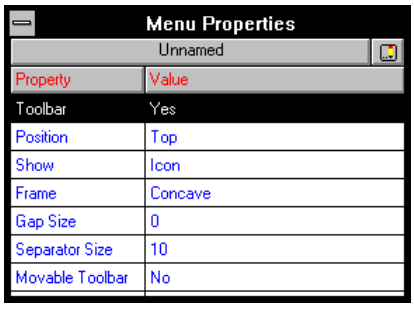
   The Menu Editor window appears with the cursor positioned at the empty menu item:



   The Menu Item Properties window displays the properties associated with this item.

2. Click on the Pencil icon in the Menu Item Properties window.

   The Menu Properties window now displays the general properties for the menu, as indicated by the title bar:



3. Type **CustomerMenu** in the edit control and press Enter.

   This assigns a name to the menu. The menu name is used to generate the menu class for your menu and is also used to attach your menu to a window.

Using Auto Layout

The easiest way to define a menu is to use the Auto Layout feature of the Menu Editor. This feature allows you to add predefined menus that are commonly found in Windows applications. It also fills in many of the Menu Item Properties for each of the predefined menus.

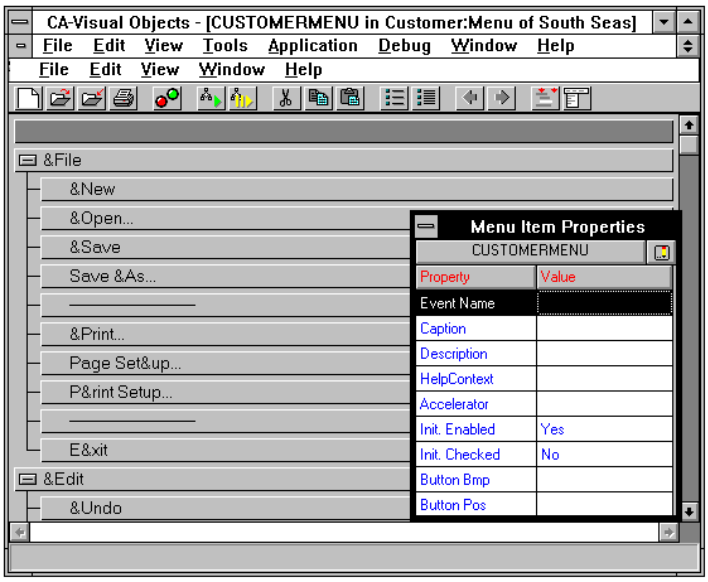
1. Select the Auto Layout toolbar button.

   The Auto Layout dialog box appears:

   Auto Layout allows you to choose from the five menus that are most often seen in Windows applications: File, Edit, View, Window, and Help. The selected menu items are appended to the menu you are currently defining.

2. Choose OK to accept all the predefined menus.

The Menu Editor window appears:



The menu hierarchy is displayed in a tree-like structure. The first level of items are those displayed on the menu bar. The indented items are the commands that are associated with the corresponding menu. For example, the File menu has the New, Open, Save, Save As, Print, Page Setup, Print Setup, and Exit commands.

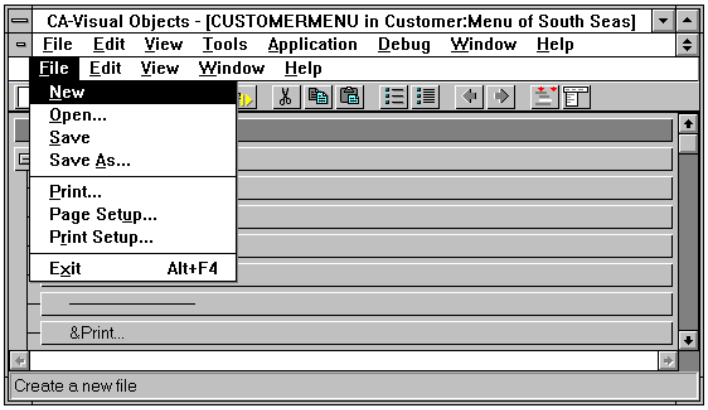
Previewing Your Menu

You may have noticed that a second menu bar appears on the Menu Editor window, once you have created menu entries using Auto Layout, as seen below:

This menu, directly above the toolbar is the *preview menu bar*. The preview menu bar not only displays the menu you are creating, but it is a working prototype of the new menu.

Here is an example of how the preview menu bar is useful when developing your application's menu:

1. To preview the File menu from the Menu Editor, click on File in the preview menu bar. The pull-down menu associated with the File menu appears, just as it would appear in your application:



2. You can also click on some of the other menu entries of the preview menu bar to see how they appear in the application.

Collapsing/Expanding the Menu Structure

The menu structure you are currently viewing is quite long. Here are some ways to make the structure easier to view and manipulate:

1. You can control the display of a single branch of the tree by clicking on its Collapse (⊟) or Expand (⊞) toggle button.

2. Simplify the display at any time by selecting the Collapse All toolbar button.

   This collapses all branches of the tree. The Expand button is now present to the left of all menus with menu items.

3. To return to the full display, select the Expand All toolbar button.

   The Collapse button is now present next to each item.

Adding an Item to the Hierarchy

Let's add a Report menu, which contains a command for the Customer List Report, to the predefined entries created using Auto Layout:

1. Scroll down through the Menu Editor and click on the View menu.
2. Press Enter to create an empty item ***after*** the View menu.
3. Type **&Report,** and press Enter.

   This creates the Report menu and inserts an empty item below it.
4. Type **&Customer List...**

   Note: Do ***not*** press Enter.

Tip: Each menu and menu item in a Windows application typically features a single underlined letter that indicates how to select it from the keyboard. For example, the "R" in the Report menu and the "C" in Customer List menu entry are underlined, indicating that you can select the Customer List Report command by pressing the Alt+R, C key combination. To add this type of functionality to your menus, simply preface the letter that is to be underlined with an ampersand (&).

5. You can preview the modified menu items and their key combinations, by clicking on the preview menu bar.

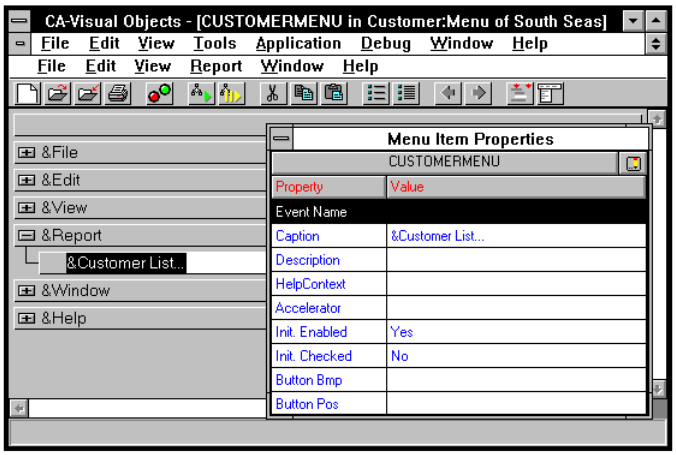
Changing the Hierarchy of a Menu Item

You now have two new entries on the same hierarchical level. The Customer List should be a child of the Report menu. You must now redefine the hierarchy between the two new items as follows:

1. Click on the Customer List menu item, to make it the currently selected entry.

2. Select the Demote Item toolbar button.

   This makes Customer List a menu item under, or a child of, the Report menu:



3. Preview the modified hierarchy by selecting the Report menu from the preview menu bar.

Removing Menu Items from the Hierarchy

Since you do not need all the menu items created by Auto Layout, let's remove a few:

1. Select the Save As menu item, under the File menu, in the Menu Editor. (If you have collapsed your menus, expand them to select the Save As menu item.)
2. Delete the item, by selecting the Delete Item command from the Edit menu (or by pressing Ctrl+Y).
3. Using the two previous steps, delete the following menu items:
   - From the File menu: Save, Print, and Page Setup
   - From the Edit menu: the first or second separator line, Insert, Delete, Go To, and Find Next
   - From the View menu: the separator line, All Records, and Select Records
   - The empty menu item, above the File menu, that was initially created as the first item in the Menu Editor

Specifying Menu Actions to Perform

For a menu item to perform an action in your application, you must specify an Event Name in the Menu Item Properties window. If you review the menu items you created with the Auto Layout feature, you notice that each contains an Event Name in its Menu Item Properties window.

The Event Name specifies the name of a method to call, which the menu attempts to locate in several ways. First, the menu attempts to call this method from within its owner window. If its owner window has the method defined or is subclassed from a window that has it defined, the method is called. If no method exists in the owner window, it attempts to call it from its owner window's owner. It continues to call all owners until it reaches the application level.

If none of the owners within the window ownership hierarchy have this method defined, the menu searches for a Window class of the same name. If one is located, the menu opens the method.

If a Window class is not found, it then searches for a ReportQueue class of the same name. If one is found, it is executed.

If a ReportQueue class is also not found, the MenuCommand() method (a Windows callback) provides default behavior so that your applications can be prototyped and compiled successfully during development.

Note: See the GUI Classes topic, in CA-Visual Objects online Help, for a complete description of event handling.

The fact that nothing happens (which also means there is no runtime error) makes CA-Visual Objects a great prototyping tool. You can define your menus up front and make them functional incrementally.

Now, let's use a method of the South Seas Adventures SSAWindow as the Event Name for the Customer List command:

1. Select the Customer List menu item, under the Report menu, in the Menu Editor.
2. Click on the Event Name property of the Menu Item Properties window.
3. Type **CustomerReport** in the Value column.

   This particular menu is to be attached, later in this lesson, to the EditCustomerWindow. The EditCustomerWindow does not have a CustomerReport() method. However, the EditCustomerWindow's owner, SSAWindow, does. It is this method that is invoked when the Customer List menu item is selected.

Providing Menu Short Cuts

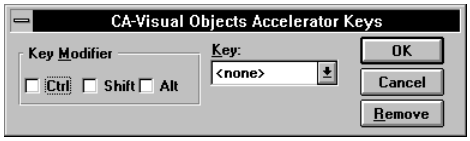
You may further simplify your application, by providing an *accelerator* (or shortcut) key. An accelerator key is usually attached to frequently used menu commands. It allows your user to press a key combination and directly access a command, bypassing the menu altogether.

To provide this functionality, you need to specify a key, or a key combination, in the Menu Item Properties window of a menu item. CA-Visual Objects automatically appends the name of the accelerator key to the menu command caption.

Let's define Ctrl+F10 as the accelerator key for the Customer List Report command:

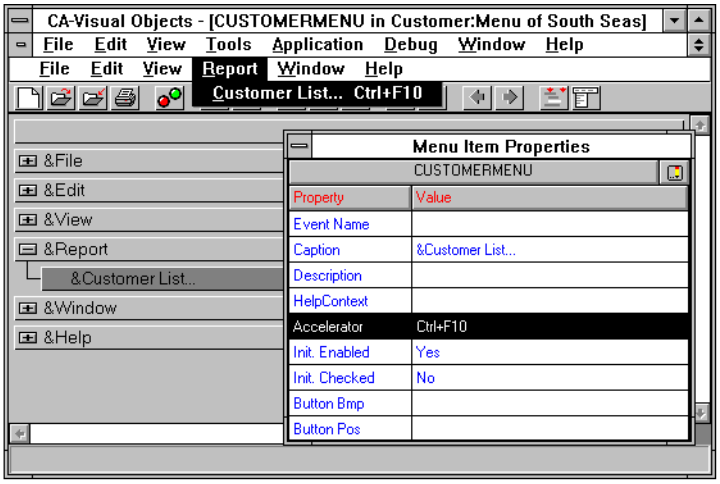
1. Select the Customer List menu item, under the Report menu, in the Menu Editor.
2. In the Menu Item Properties window, click on the Accelerator property.

   The CA-Visual Objects Accelerator Keys dialog box appears:



3. Click on the Ctrl option in the Key Modifier group and select F10 from the Key combo box.
4. Choose OK.

To view the accelerator key for the Customer List menu item, select the Report menu from the preview menu bar, as seen below:

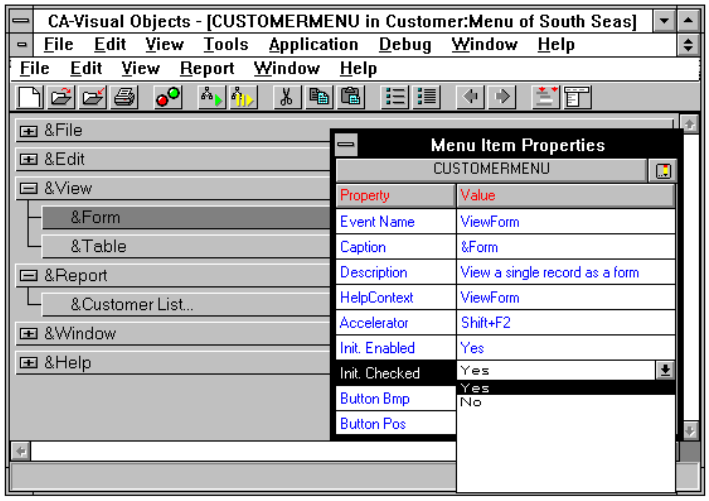


Checking a Menu Item

Your menu can be defined to provide a visual cue for a current selection—by displaying a check mark next to your menu command. For example, the menu you are currently defining is to be attached to a window that initially displays data in form view. You can place a check mark next to the Form menu item, to indicate the current view state for your user, by following these steps:

1. Select the Form menu item, under the View menu, in the Menu Editor.
2. In the Menu Item Properties window, click on the Init. Checked property.

3. Select Yes from the list box in the Value cell, as seen below:



Later in this lesson, when a different view is selected, the check mark from the Form View command is removed.

Creating a Toolbar

As stated earlier, toolbars are attached to windows, yet you create them using the Menu Editor. The reason for this is CA-Visual Objects associates the buttons of a toolbar with menu commands used in the application.

Typically, only a few menu commands are depicted on the toolbar—those which are most frequently used.

When you originally create a menu, an empty toolbar is associated with this menu. You may preview the default toolbar associated with the menu by following these steps:

1. Select the Preview Toolbar command, from the File menu of the Menu Editor. (Be sure to make your selection from the CA-Visual Objects menu and not from the preview menu bar.)

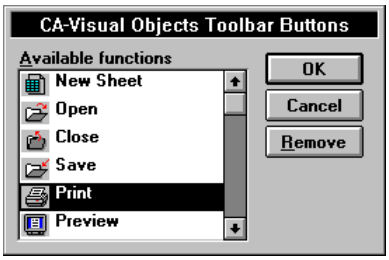
   The Toolbar Preview window appears, showing an empty toolbar.

2. Double-click on the Menu Editor Toolbar Preview window's system menu, to return to the Menu Editor.

To create toolbar buttons, you simply need to define the Button BMP property for the menu items to be accessed from the toolbar:

1. Select the Customer List menu item, under the Report menu, in the Menu Editor.
2. Click on the Button BMP property in the Menu Item Properties window.

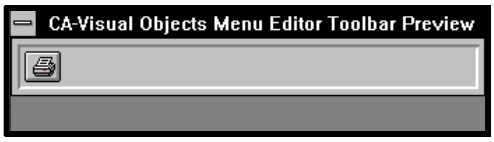
   The Toolbar Buttons dialog box appears:



3. Select the Print icon from the list and choose OK.

   The Print icon is now displayed in the Value cell along with the caption that appears on the status bar when the user moves the cursor over the icon. On the next line, in the Menu Item Properties window, the Button Pos value is now 1. This indicates that this is the first button on the toolbar.

4. Select the Preview Toolbar command from the Menu Editor's File menu, to preview the toolbar you are creating:



5. Double-click on the Menu Editor Toolbar Preview window's system menu, to return to the Menu Editor.
6. Select the Next menu item, under the Edit menu.
7. Click on the Button BMP property in the Menu Item Properties window.
8. Scroll down through the list of available options (near the bottom) to select the Next Record icon  from the Toolbar Buttons dialog box and then choose OK.

The Button Pos value is 2, indicating that the Next Record icon is the second button on the toolbar.

9. Select the Previous menu item, under the Edit menu.
10. Click on the Button BMP property in the Menu Item Properties window.
11. Scroll down through the list of options (once again, near the bottom) to select the Previous Record icon from the Toolbar Buttons dialog box and then choose OK.

    The Button Pos value is 3, indicating that the Previous Record icon is in the third button on the toolbar.
12. Select the Form menu item, under the View menu.
13. Using steps 10 and 11, assign the View Form icon to the Button BMP.
14. Select the Table menu item, under the View menu.
15. Assign the Table icon to the Button BMP. Find this bitmap by scrolling down through the list (do not use the View Table bitmap).
16. Select the Preview Toolbar command from the Menu Editor's File menu to preview the completed toolbar:

17. Double-click on the Menu Editor Toolbar Preview window's system menu to return to the Menu Editor.

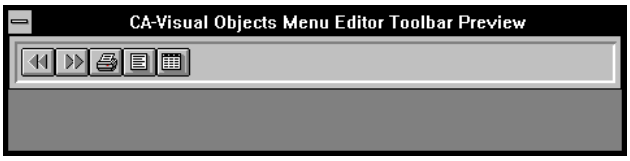
Changing Toolbar Button Positions

You may reorder the position of toolbar buttons, once created, by changing the Button Pos value in the Menu Item Properties window.

1. Select the Customer List menu item, under the Report menu.
2. Click on the Button Pos property in the Menu Item Properties window and change the value to **3**.

   This moves the Printer button assigned to this command to the third position.
3. Select the Previous menu item, under the Edit menu.
4. Click on the Button Pos property in the Menu Item Properties window and change the value to **1**.

   This moves the Previous Record button assigned to this command to the first position.
5. Select the Preview Toolbar command from the Menu Editor's File menu to preview the modified toolbar:



6. Double-click on the Menu Editor Toolbar Preview window's system menu to return to the Menu Editor.

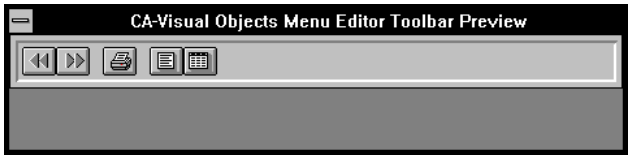
Spacing Between Toolbar Buttons

You may also modify the gaps, between buttons or a group of buttons, for your toolbar. This is achieved by skipping a number when assigning button positions, detailed below:

1. Select the Customer List menu item, under the Report menu.
2. Click on the Button Pos property in the Menu Item Properties window and change the value to **4**.

   This leaves a gap before the Printer button.
3. Select the Form menu item under the View menu.

4. Change the Button Pos to **6** to leave a gap before the ViewForm button.
5. Select the Table menu item under the View menu and change the Button Pos to **7**.
6. Select the Preview Toolbar command from the Menu Editor's File menu to preview the modified toolbar:



   Notice the new icon positions and the gaps before and after the third icon.

7. Double-click on the Menu Editor Toolbar Preview window's system menu to return to the Menu Editor.

Other Modifications to the Toolbar

In addition to changing button positions and modifying the spacing between buttons, toolbars have several other properties that can also be modified.

Let's change the frame property and the separator size for the toolbar that you created:

1. Click on the pencil icon in the Menu Item Properties window (or select the Menu Properties command from the Edit menu).

   The Menu Properties window is displayed.

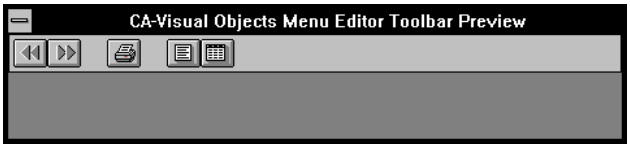
2. Click on the Frame property and select Plain from the list box.

   This changes the appearance of the frame around the toolbar.

3. Click on the Separator Size property and type **20** in the value cell.

   This increases the separator spacing between the toolbar entries.

4. Select the Preview Toolbar command from the Menu Editor's File menu to preview the modified toolbar:



   Notice the different background for the toolbar and the increased distance before and after the third icon.

5. Double-click on the Menu Editor Toolbar Preview window's system menu to return to the Menu Editor.

   For a description of other Menu Properties, see Using the Menu Editor topic in CA-Visual Objects online Help.

Saving the Menu

When you have completed your menu definition, you must make it available to the Window Editor. To do this, you simply:

1. Save the menu by clicking on the Save toolbar button.

2. Select the Build toolbar button.

3. Close the Menu Editor by double-clicking on its system menu.

Attaching a Menu to a Data Window

To use the menu you have created, you must attach it to a window. You are going to attach the CustomerMenu to the EditCustomer window, created in the "Creating and Using Windows" chapter, as follows:

1. Open the Customer:Forms module by double-clicking on its button in the Module Browser.
2. Open the EditCustomerWindow window by double-clicking on this entity from the Entity List which appears.
3. In the Data Window Properties window, scroll down through the properties and select the Menu property by clicking on it.

4. Replace the SSAChildMenu with the CustomerMenu from the combo box.

5. Save your changes by selecting the Save toolbar button.

6. Close the Window Editor by double-clicking on its system menu.

Because the EditCustomerWindow has a different menu attached to it, you will have to make a change to its ViewToggle() method, which is in the Tutorial:Windows module. This module was imported in the "Creating and Using Windows" chapter as TUTWIND.MEF.

The ViewToggle() method is one of the few developer-coded menu event methods. It modifies the state of menu items by using the CheckItem() and UncheckItem() methods. These methods require you to specify the name of a constant, such as IDM_SSACHILDMENU_VIEW_TABLE_ID, to identify the menu item. Since the EditCustomerWindow now utilizes the new CustomerMenu, these constants must be changed. To do this:

1. Open the Tutorial:Windows module by double-clicking on its button in the Module Browser.
2. Find the EditCustomerWindow:ViewToggle() method and double-click on its button to open the Source Code Editor.
3. Select the Edit Replace command.
4. Type SSACHILDMENU in the Find What edit control and type CUSTOMERMENU in the Replace With edit control.
5. Choose Replace All.

   You will see that the new substring appears in several different lines.
6. Double-click on the system menu and select Yes to save the changes.

7. Rebuild the application by selecting the Build toolbar button.

Putting it All Together

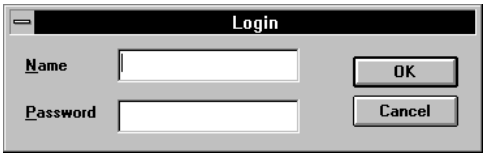
You are now going to run and test the application with the changes you have made:

1. Make sure the application has been rebuilt—such that all modules in the Customer:Forms Entity Browser have a green LED. If not, rebuild the application by selecting the Build toolbar button.

2. Run the application by selecting the Execute toolbar button.

3. Choose OK at the opening dialog box.

   The Login dialog box appears:



4. Type **User** in the Name edit control and **Trainee** in the Password edit control. Choose OK.

5. Select the Open command from the File menu.

6. Click the Customer radio button and choose OK.

   The Customer Browser is displayed.

7. Select a customer by clicking on one of the cells in the Customer Browser and choose Edit. (This functionality was added in the "Creating and Using Windows" chapter.)

   The Edit Customer window is displayed with the menu and toolbar you created for it.

8. View your modified menu by selecting the Report and/or the Edit menu.

9. Close the application by double-clicking on its system menu.

Designing a Menu

When designing your application, you face a decision about the number of menu definitions that are required. One possible approach is to create a menu for each window of your application.

However, in many applications you find that most actions required in windows are common to many windows. Therefore, defining a separate menu for each window is redundant. Another approach is to find common functionality between your menus and to manage the exceptions.

The SSAChildMenu

Before you created the CustomerMenu menu, you may have noticed that the South Seas application uses only one menu for all the child windows.

Since most windows in the South Seas Adventures application fall into one of two categories—the Edit or New windows—only one menu (SSAChildMenu) has been created to be used by most data windows. The actions defined in this menu are generic enough to be used and unchanged by each Edit window. As for the New windows, some of the commands (such as “movement” commands) have been disabled.

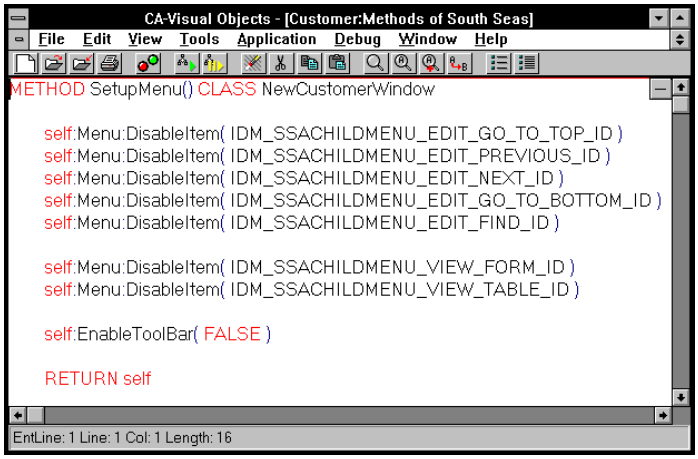
Customizing a Menu

In certain instances, you may want to further customize your menu. For example, the NewCustomer window does not require that movement in the data server be allowed while a customer is being added. It also does not allow switching between Form and Table view. In addition, since most toolbar buttons are movements, the toolbar is disabled as well.

Disabling Menu Items

The SetUp() and SetUpMenu() methods for the NewCustomer window have been created to make the desired menu modifications.

You can find the source code for the SetupMenu() method in the Customer:Methods module of the South Seas Adventures application, as shown below:



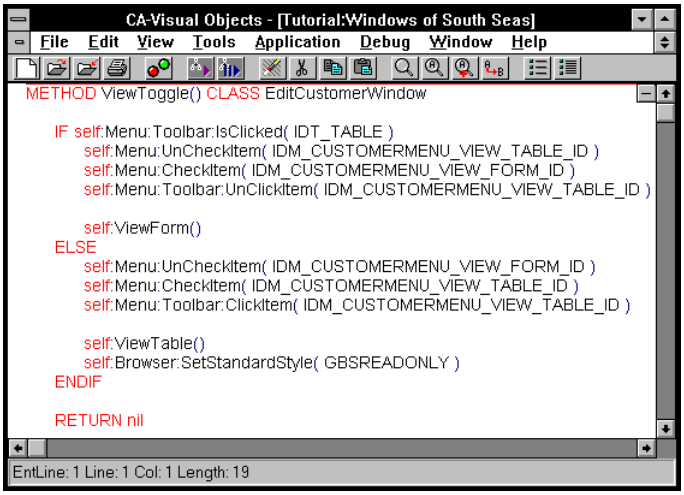
Note: When a NewCustomer window is displayed, the disabled items appear dimmed on its menu.

Editing Toolbar Buttons

You can modify the standard behavior of toolbar buttons using methods of the menu class. For example, the EditCustomer window can be modified to use the same toolbar button to toggle the Form/Table views.

When the Table view is selected from the menu command or with the button, the button is displayed with a depressed shape. This is accomplished with the ViewToggle() method of the EditCustomerWindow class. (This method is contained in the Tutorial:Windows module of the application.)

The ViewToggle() method is shown below:

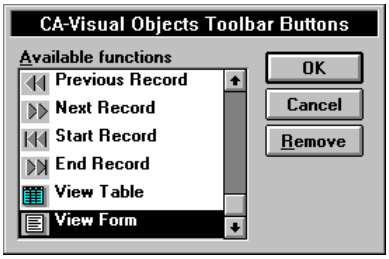


To use this method in your menu, you change the ViewTable event name from ViewTable to ViewToggle and remove the ViewForm button from the toolbar. To do this, you need to go back to the menu you created earlier in this lesson:

1. Double-click on the Customer:Menu module button in the Module Browser.
2. Invoke the Menu Editor by double-clicking on the CustomerMenu menu entity in the Entity Browser.
3. Select the Table menu item, under the View menu.
4. In the Menu Item Properties window, select the Event Name property and type **ViewToggle**.
5. Select the Form menu item, under the View menu.

6. In the Menu Item Properties window, click on the Button BMP property.

   The Toolbar Buttons dialog box is displayed with the View Form icon highlighted:



7. To remove the View Form button from the toolbar, choose Remove.

8. Save your changes by selecting the Save toolbar button.

9. Build the application by selecting the Build toolbar button.

To view the results of your changes, follow the steps described earlier, in Putting it All Together, to execute the application and access the EditCustomer window.

Summary

In this chapter, you have learned how to use the Menu Editor to create menus and toolbars, change a menu and its menu item properties, prototype menus and toolbars, and define accelerator keys. You have also seen how to modify the behavior of menus by programming new behavior.

In the following lesson, you will see how to access data (from controls, data servers, and data windows) and gain an understanding of how data is processed in CA-Visual Objects.