Maximizer™
The CRM Company

MaxForms

Maximizer Form Designer

User's Guide

Contents

CHAPTER 1

Introduction

Introducing Maximizer Form Designer

In this chapter...

Welcome to Maximizer Form Designer

Maximizer Form Designer provides users with the ability to display Maximizer Address Book entries via custom forms, in addition to the traditional Maximizer method of displaying data in list-based views.

Maximizer Form Designer is a standalone application specifically designed for creating custom forms for Maximizer. It has its own Help file, which includes summaries of procedures and reference information on the Properties, Methods and Events applicable to the various objects you can place on a form.

Naturally, certain objects can display or otherwise interact with fields from a Maximizer Address Book folder. These include the EditBox object (which is really a standard field), a Date and Time picker (for date/time fields) and a Masked Edit object which facilitates the inclusion of information such as telephone numbers or credit card details, which might require the data to conform to a specific format.

Once a custom form has been created, it is available within Maximizer. If you click on the Custom Forms button on the Maximizer toolbar, you'll see a list of forms. Initially, this list includes a small selection of supplied custom forms that you can use. Any forms you create and save yourself will be displayed in this list as well.

Using Maximizer Form Designer

Designing a form is a simple “drag-and-drop” process. In the Form Designer editor, you can select a design object and drop it into your form. Design objects are “data aware”, which allows them to display and save Maximizer data, and with the built-in scripting language, you can customize your form’s functionality.

Form Design Objects

The design objects allow you to create powerful forms to represent the data stored in your Maximizer Address Book folder. You can use the objects only within the Maximizer environment.

Some objects available in the Form Designer include the Button object, the ListBox object, the Tab Control, the Tree Control, and various edit boxes. Maximizer Form Designer allows you to create forms using most standard form controls.

Data awareness

Maximizer Form Designer objects are “data aware”, meaning they connect to, interrogate and save data in the Address Book folder in which Maximizer data is stored.

Certain objects, such as the EditBox object, are fields which have the capability of displaying Maximizer data. To achieve this, simply place an EditBox object on the form, open its property sheet (by right-clicking on it or selecting View, Properties) and set the Maximizer Field property to the desired field (a list is read from the currently-open Maximizer Address Book folder).

Smart Forms

Also included is the VB scripting language, Enable Script, which allows you to employ the objects in a fully programmable environment. With or without Enable Script, it’s now possible to create impressive forms in Maximizer.

Running Maximizer Form Designer

After installation, Maximizer Form Designer can be run from within Maximizer or from your Windows Explorer.

➤ To run Maximizer Form Designer

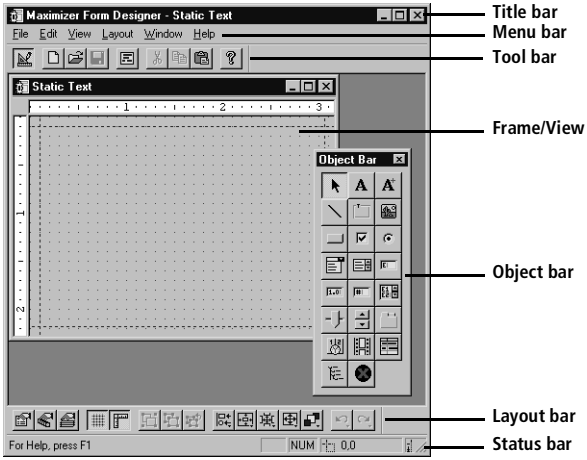
- Within Maximizer, choose Tools > Maximizer Form Designer.

  – or –

- In Windows Explorer, double-click on the file Maxforms.exe in your Maximizer program folder.

Getting to Know the Form Designer Screen

The Maximizer Form Designer screen contains several elements to assist you in the creation of forms.



- The Title bar displays the name of the program (i.e., “Maximizer Form Designer”) as well as the title of the form that you are currently editing.
- The Menu bar provides keyboard and mouse access to all functions of the Form Designer.
- The Toolbar is displayed across the top of the application window, below the menu bar. The toolbar provides quick mouse access to many tools used in the Maximizer Form Designer.
- The Frame/TheView is the term used to describe the workspace in which forms are designed. In design mode, you can right-click on the TheFrame/TheView to bring up its property sheet, in which you can modify properties such as the Title and Background color, enable or disable scrollbars, as well as others. Please see the section entitled, “The Frame/The View” in the “Objects” chapter for more information on setting these properties.
- The Object bar contains 22 buttons for design objects. Click on an object button in the Object bar to choose which object you would like to add to your form.
- The Layout bar contains several tools for the alignment, spacing, and grouping of objects. Other buttons open the Object Sheet, Property Sheet, or Layers Sheet, or turn on and off other design elements.

- The Status bar is displayed at the bottom of the Maximizer Form Designer window. To display or hide the status bar, use the Status bar command in the View menu.

The left area of the status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as you depress them, before releasing them. If after viewing the description of the toolbar button command you wish not to execute the command, then release the mouse button while the pointer is off the toolbar button.

The right areas of the status bar indicate which of the following keys are latched down:

Indicator	Description
CAP	The Caps Lock key is latched down.
NUM	The Num Lock key is latched down.
SCRL	The Scroll Lock key is latched down.

CHAPTER 2

Working with Forms

Create Custom Forms for use in Maximizer

In this chapter...

Creating and Editing Forms

The editor in Maximizer Form Designer is designed for easy "drag-and-drop" form editing. At design time, you can add controls to your form, edit the layout of the form, add VBScript to objects, and more.

The following list shows some of the basic procedures you should become familiar with when you create a form.

Action	Description
Create and edit a form	Creating a new form; and modifying an existing form.
Add objects to a form	Showing the Object bar; alternative methods for adding objects; and importing controls from external windows.
Place and size objects	Selecting, moving, deleting or copying objects; and sizing individual objects.
Arrange objects	Using Snap to Grid; aligning, spacing and resizing objects.
Change the tab order	Setting and changing the tab order for form controls.
Change the dominant object	Using Ctrl+click to change the dominant object.
Defining shortcut keys	Defining the accelerator key (or "mnemonic") for an object; and checking for redundant accelerator keys.
Inspecting objects	The Object Sheet provides a list of all objects, which can then be further manipulated.
Testing a form	Switching from edit mode to run mode to test a form, or using preview mode.

Starting a New Form from Scratch

➤ **Starting a new form**

1. Run Maximizer Form Designer.
2. From the File menu, choose New.
3. You are presented with a form workspace and the Object bar.

Modifying an Existing Form

➤ **To open an existing form**

- From the File menu, choose Open.

Adding Objects to a Form

➤ **To add objects to a view using point and click**

1. Run Maximizer Form Designer.
2. On the Object bar, click the button for the object you want. If you are not sure which object you want to place on the form, hold your mouse pointer over a button for a moment and the name of the object will appear (this feature is called a "tooltip").
3. Move the mouse pointer to the frame and click at the position you want. The object is given a default size as previously defined for the type of object.

➤ To add objects to a view by dragging with the mouse

1. On the Object bar, click the button for the object you want.
2. Place the mouse pointer where you want the upper left corner of the object to be located.
3. Hold down the left mouse button.
4. Drag the mouse pointer down and to the right; a dotted outline of the object appears.
5. When the object is the size you want, release the mouse button.

Once an object has been placed on the form, you can right-click on it to bring up a list of its properties. From the property list you can modify any setting relating to the object.

Selecting Objects

When you are sizing or aligning multiple objects, Maximizer Form Designer uses the "dominant object" to determine how the other objects are sized or aligned. When multiple objects are selected, the dominant object has normal sizing handles; all the other selected objects have a hatched border without sizing handles. The dominant object is the first object selected, unless it is reset by holding down the Ctrl key while clicking with the mouse.

➤ To select an object

- Point to the object you want and click. The selected object (object or view) is deselected.

  – or –

- Use Tab to move forward or Shift+Tab to move backward through the objects in the frame.

➤ To select more than one object

You can select multiple objects by clicking them individually while holding down the Shift key. In addition, clicking a selected object with the Shift key held down deselects it. Once you have selected one or more objects, you can remove or add individual objects without disturbing the selection as a whole.

1. From the Object bar, make sure the pointer tool is selected.
2. Hold down the left mouse button and drag to draw a selection box around the objects you want to select. Objects partially outside the selection box are not selected.
3. Release the mouse button; all objects inside the selection box are selected.

➤ To remove from or add to an existing selection

- Hold down the Shift key and click the object you want to remove from or add to the existing selection.

Moving Objects

You can use the following procedures to move one or more objects from one location to another in a view, or from one view to another. If Snap to Grid is on, the object(s) snaps to the alignment grid when moved by the mouse. For information on other ways to align multiple objects, see Aligning Objects.

➤ To move an object from one location to another in a view

- Drag the object to its new location.

  – or –

- Select the object and use the arrow keys to move the object one pixel at a time. Hold down the Ctrl key to move one grid unit at a time.

➤ To move an object from one view to another

- Use the Edit menu's Cut (Ctrl+X) and Paste (Ctrl+V) commands. The object is placed in the same position as in the original view.

  – or –

- Shortcut: Ctrl+X (Cut) and Paste (Ctrl+V)

Aligning Objects

➤ To align objects

1. Select the objects you want to align.
2. Make sure the correct dominant object is selected. The final position of the group of objects depends on the position of the dominant object.
3. Choose one of the following tools on the Layout bar:
   - Align Left—aligns the selected objects along their left side. (Ctrl+LEFT ARROW).
   - Align Right—aligns the selected objects along their right side. (Ctrl+RIGHT ARROW).
   - Align Top—aligns the selected objects along their top edges. (Ctrl+Up Arrow).
   - Align Bottom—aligns the selected objects along their bottom edges. (Ctrl+Down Arrow).

➤ To align objects on their center

1. Select the objects you want to center.
2. Make sure the correct dominant object is selected. The final position of the group of objects depends on the position of the dominant object.
3. From the Layout menu, choose Align Objects > Vert. Center, or Align Objects > Horz. Center.

➤ To center objects in the frame

1. Select the object or objects you want to rearrange.
2. Choose one of the following tools on the Layout bar:
   - Center Vertical—centers objects vertically in the dialog.
   - Center Horizontal—centers objects horizontally in the dialog.

Sizing Individual Objects

Use the sizing handles to resize an object. When the mouse cursor is positioned on a sizing handle, it changes shape to indicate the direction in which the object will be resized. Active sizing handles are solid; if a sizing handle is hollow, the object cannot be resized along that axis.

When you change the size of an object, its final shape may be affected by whether you have Snap to Grid turned on.

➤ To resize an object

1. Click the object or select it with the Tab key.
2. Use the sizing handles to change the size of the object:
   - Sizing handles at the top and sides change the horizontal or vertical size.
   - Sizing handles at the corners change both horizontal and vertical size.

– or –

Use the Shift key plus the arrow keys to resize the object one pixel at a time, or the Ctrl-Shift keys plus the arrow keys to resize the object one grid unit at a time.

Spacing Objects

➤ To space objects evenly either down or across

1. Select the objects you want to resize.
2. From the Layout menu, choose Space Evenly > Across, or Space Evenly > Down.

Copying Objects

➤ To copy an object

- Use the Edit menu's Copy and Paste commands.

  – or –
- Shortcut: Ctrl+C (Copy) and Ctrl+V (Paste)

When you paste an object into a new view, it is placed in the same position as it was in the old view. You can use copy and paste to copy controls from one form to another in the editor.

Deleting Objects

➤ To delete an object

1. Select the object.
2. From the Edit menu, choose Cut or Delete.

   – or –

   Shortcut: Ctrl+X (Cut) or DEL (Delete)

Testing a Form

By switching out of edit mode into run mode, you can test the behavior of a form. This gives you immediate feedback on how the layout of objects appears and performs, and speeds up the user interface design process.

➤ To test a form

- Toggle the Edit item in the Layout menu.

  – or –

  Toggle the Edit button in the Layout bar (if available).

  – or –

  Shortcut: Ctrl+E

In preview mode a run-only copy of the form is created. This is useful for testing the form without saving the file.

➤ To preview a form

- Select the Preview Mode item in the Layout menu.

Using Snap to Grid

When you are placing or arranging objects in a view, you can use the alignment grid for more precise positioning. When the grid is turned on, objects appear to “snap” to the dotted lines of the grid as if magnetized. You can turn this “snap to grid” feature on and off, and change the size of the grid cells.

➤ To turn Snap to Grid on or off for the currently active view (in edit mode)

1. On the Layout bar (if available), click Snap to Grid.

   – or –

   From the Layout menu, choose Grid Settings. The Grid Settings dialog box appears.

2. Toggle the Snap to Grid checkbox.

   – or –

   Shortcut: Ctrl+G

➤ To change the size of the layout grid

1. From the Layout menu, choose Grid Settings. The Grid Settings dialog box appears.
2. Enter the height and width in pixels for the cells in the grid.

Using Line Snapping

Objects can be permanently connect by line snapping. When they are automated moved, the lines are moved to maintain the connections. The endpoints of lines are "snap ties" which are matched up with "anchor points" in other objects. Built-in objects with anchor points include Text, TextVar and Bitmap.

Objects with anchor points also have the "AnchorSnaps" property, which brings up the dialog, above. An unlimited number of anchor points can be defined. Each is defined by a pixel offset (by default by 0,0) from one of eight base points around the perimeter and one base point in the middle of the object.

Adding Labels to Your Form

These instructions apply to adding labels while you are in Design mode in Maximizer Form Designer. Labels properties may also be applied while you are Run mode. For more information, please read the following instructions on Adding Labels in Run mode.

➤ To add a label to your form

1. From the Object bar, select the Text tool. Alternatively, if you are assigning a variable to the label, you can select the TextVar tool.
2. Move your mouse to the position you would like to place the label and click the left mouse button.
3. Click anywhere in the text and clear the "Text".
4. Type your label.
5. Double-click within the text box. The Property Sheet appears.

6. Assign the label's properties and/or event controls.
7. Close the Property Sheet. This makes your changes to the form. When you run the form or attempt to close it, you are prompted to save your changes.

➤ To add a label to your form in Run mode

1. From the Object bar, select the Text tool. Alternatively, if you are assigning a variable to the label, select the TextVar tool.
2. Move your mouse to the position you would like to place the label and click the left mouse button.
3. Click anywhere in the text and clear the "Text".
4. Type your label.
5. Double-click within the text box. The Property Sheet appears.
6. Click the ellipsis button, which is located on the Events tab next to the EventClick property. The VBS Mini-Editor appears.
7. Insert your script. To access the available properties in Run mode, you must type the name of your label followed by a period. (.) This activates the VBS Mini-Editor.
8. Click OK to close the editor and then click the close button to close the Property Sheet. This makes your changes to the form. When you run the form or attempt to close it, you are prompted to save your changes.

Assigning Keyboard Access to Controls

Normally keyboard users move the input focus from one control to another in a view with the Tab and arrow keys. However, you can define a mnemonic key that allows users to choose the control by pressing a single key.

For controls with no visible caption, you may wish to add a Text object as a label. In this case, be sure to assign the same mnemonic character (using an ampersand) to the Text object as you are using for the control it labels.

➤ To define a mnemonic key for a control with its own visible caption (pushbuttons, check boxes, and radio buttons)

1. Select the control and open the Properties sheet.
2. In the Text property, type an ampersand (&) in front of the letter you want as the mnemonic for that control. An underline appears in the displayed caption to indicate the mnemonic key.

➤ **To define a mnemonic key for a control without its own visible caption (edit boxes, combo boxes, list boxes, spinner, and slider controls)**

1. Select the control and open the Properties sheet.
2. Click on the Accelerator property to open a drop list.
3. Select the alphanumeric character that you wish to use as the mnemonic key.

➤ **To check for duplicate mnemonics, choose Check Mnemonics from the Layout menu.**

You are warned of any conflicts via a message box, and given the opportunity to select the group of objects which have the mnemonic key conflict.

Setting and Changing the Tab Order

Tab order is the order in which the tab key moves the input focus from one object to the next within a view. To make a object part of the tab order, set the Tabstop property to YES. Non-control objects do not have the Tabstop property and can not be part of the tab order.

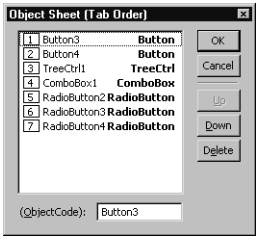
If your view contains overlapping objects, changing the tab order may change the way the objects are displayed. Objects which come first in the tab order are always displayed on top of any overlapping objects and non-control objects that follow them in the tab order.

➤ **To change the tab order for a particular object on a form**

1. Select the object.
2. From the Layout menu, choose Set Tab Order. A number at the upper left of the control shows its current tab order.
3. Change the tab order by clicking inside the control or by pressing the Tab or Shift-Tab keys.
4. Press Enter or click the mouse outside the control to exit Set Tab Order mode.

Using the Object Sheet

The Object Sheet dialog allows the objects in a view to be listed rather than viewed.



The ordering of an object can be moved using the Up or Down buttons, which affects drawing order or tab order for controls. Objects are displayed in blue text, while controls are in black text with their tabbing sequence number at left.

Viewing the Properties and Methods for an Object

Once you have added an object to a form in Maximizer Form Designer, you may view all of its available properties and methods.

➤ To view the properties and methods for an object

1. Add an object to a form, if you haven't already done so.
2. Double-click or right-click inside the area designated for the object on the form. The Property Sheet appears.
3. Name the object. This is specified in the ObjectCode field on the Normal tab of the Property Sheet.
4. Click the Events tab.
5. In the second column of any of the Events fields, click your left mouse button. The VBS Mini-Editor appears.
6. In the editor text box, type the name of the object, followed by a period (.). For instance, if you have named the form Tab1, type **Tab1.** It's important to note that this text is case sensitive, so you must be precise in using the exact name of the object.

Immediately after entering the text, a scrolling list appears displaying all of the available properties and methods for the selected object. Events are denoted by a green colored icon and properties by an icon that resembles a hand holding a properties sheet.

Changing the Dominant Object

When you are resizing or aligning multiple objects, the Form Designer editor uses the dominant object to determine how the other objects are sized or aligned. When multiple objects are selected, the dominant object has normal sizing handles; all the other selected objects have a hatched border without sizing handles.

➤ To change the dominant object when more than one object is selected

- Hold down the Ctrl key and click the object you want to influence the size or location of the others. All further resizing or alignment is based on this object.

Changing Redraw Order

There are two ways to change the redraw order: repeated clicking on the overlapped portions of multiple object or choosing the Bring to Front or Send to Back menu items (which can be optionally included by the programmer).

➤ To change the redraw order for overlapped objects

1. Click the mouse repeatedly over the overlapped portion of the objects; you should see the objects change drawing order.

   – or –

   Select one or more objects from the Layout menu.
2. Choose the Layout menu's "Bring to Front" or "Send to Back".

Adding User-defined Fields to Your Forms

➤ To add a user-defined field in Maximizer Form Designer

1. From the File menu, choose Add User-Defined Field.
2. Enter a name for the user-defined field. The name must be unique in Maximizer.
3. Select the type of user-defined field you are adding. Table user-defined field items may be added in a few different ways. Please refer to the section entitled "Populating a ComboBox with User-defined Field Items" for more information.

4. Select the type of Address Book entry to which the user-defined field may be applied—Company, Individual and/or Contact.
5. Specify the attributes applicable to the chosen type of user-defined field.
6. Click OK.

Adding Bitmaps to Your Forms

To move the bitmap on the form, select it and while holding your mouse button down, drag it to the desired position. A bitmap can be moved and resized in the same way as other objects.

➤ **To add a bitmap to a form**

1. From the Object bar, select the Bitmap tool.
2. Click on the form where you would like to place the bitmap and while holding your mouse button down, drag your mouse pointer until you have the approximate size of the bitmap area.
3. Release your mouse button. The Bitmap dialog box appears.
4. Click the Load button. The Browse Files dialog box appears.
5. Locate and select the bitmap you would like to place within your form and then click Open. This places the bitmap inside the preview area of the Bitmap dialog box.
6. If you would like to specify a color for any transparent areas of the selected bitmap, choose a color from the Transparent drop-down list.
7. Click OK. This places the bitmap in your form. Note that if the size of the bitmap varies from the size of the area you have drawn with your mouse, the size is adjusted automatically.

Creating a Tab Control on a Form

A Tab control allows you to create two or more tabbed pages of controls in your form. This control can be especially useful if you need to create a form with many controls and objects, but there isn't enough space available in the standard form.

When you create a form that has a tab control on it, only the first page of the form is stored with the tab. When the user clicks subsequent tabs, the tab form will be loaded from a folder named "TabForms" under the folder in which the form resides. For example, if you design a form called "Hospital" with three tab pages, the following files are required:

```
<Maximizer path>\Forms\Hospital.mxf>
<Maximizer path>\Forms\TabForms\Hospital1.mxf>
<Maximizer path>\Forms\TabForms\Hospital2.mxf>
```

This example illustrates how the number of tabs in the control matches the number of files.

The following steps will guide you through the creation of a form with a tab control.

➤ Step 1—Setting up a folder to store secondary tabs

1. All forms are stored in a folder called Forms in the main folder where you have installed Maximizer. When using tabs in a form, the parent form (tab) is stored in this folder. All secondary tabs, or child forms (tabs) are stored in a folder called TabForms. You must create this folder yourself.
2. In the Windows Explorer, locate the folder in which you have installed Maximizer. When you install Maximizer Form Designer a folder called Forms is automatically created.
3. Open the Forms folder and create a new sub-folder called TabForms within the Forms folder.
4. When you create your child forms, save these inside the TabForms folder.

➤ Step 2—Creating your tab control form

ⓘ You can use the same objects in the all of the tabs on the form if you wish. If you do, you must rename each of the objects on the child forms with a unique name. For instance, you may want to create a tabbed form that has the same basic fields in each of the tabs in the top portion of the form and use the bottom portion of the form for the tabs that contain unique objects.

ⓘ Add the tab control *after* you have added all objects and saved the parent form.

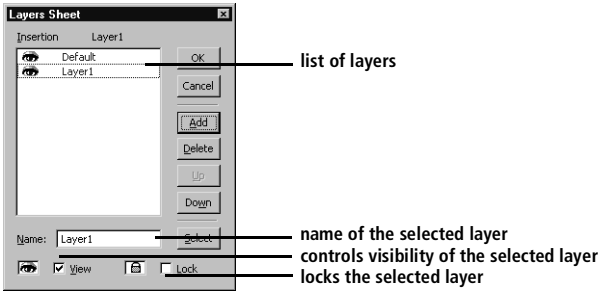
1. Design your parent form. Specify any properties and events you would like to have for the controls in the parent form. Make sure you specify a unique name for each of the controls on the form.
2. Specify the name of the form itself. To do so, click anywhere inside the form, but not inside of an object. This activates the Property Sheet. Enter the name in the FormCode field on the Normal tab.
3. From the Object bar, select the TabCtrl tool.
4. Position your mouse in the upper left section of the area where you would like the tab control to appear and while holding your mouse button down, drag the control to the full size of the tab.
5. Double-click or right-click inside the tab area while it is selected. The Property Sheet appears.
6. Click inside the second column of the ListItems property. The List Choices dialog box appears.
7. Click the Add button. This creates an item which is essentially your first tab.
8. In the name field of the List Choices dialog box, specify a name for the tab. This name appears on the tab control itself.

9 For each tab you would like to include in the form, repeat step 8. You may manually add spaces before the name of tab, in the Name field, for optimum readability. For instance, if you want the text on each tab centered, you must manually align the text with spaces.

10 Save the parent form in the Forms folder.

Creating a Form Using Layers

When you create a form with Layers, you have the ability to make objects visible or invisible upon certain events. Each control can be assigned to a specific layer, such as the Default layer, or the first, second, third and so on. At both design and run time, you can control which layers are visible.



In the Layers Sheet, you can add or remove layers, control which layers are visible in the editor, and “lock” or “unlock” layers. Once the layers are created, you assign the individual objects to particular layers using the Layer property in each object’s Property Sheet. The Lock checkbox controls a layer’s lock status; when a layer is locked, any objects assigned to that layer may not be edited.

If you create buttons that control which Layer is currently visible, you can have several “pages” of controls occupying the same physical space, while only one layer of controls is visible. One possible instance of using layers in this manner creates the illusion of a tab control. For more information on using layers to create a tab control form, see Tutorial 2—Creating a Tabbed Form Using Layers.

Creating Layers

When you start a new form in Maximizer Form Designer, only one layer exists (the “Default” layer). You can create additional layers using the Layers Sheet, which is opened by selecting Layout > Layers from the menu.

➤ To create a new layer using the Layers Sheet

When you start a new form in Maximizer Form Designer, only one layer exists (the "Default" layer). You can create additional layers using the Layers Sheet.

1. Select Layers from the Layout menu. The Layers Sheet opens.
2. Click the Add button. A new layer is added to the end of the list.
3. Enter a new name for the layer if necessary, and click OK.

➤ To make a layer visible or invisible

1. Select Layers from the Layout menu. The Layers Sheet opens.
2. Select the layer that you wish to make visible (or invisible).
3. Check the View box to make the selected layer visible, or uncheck the View box to make the selected layer invisible and click OK.

➤ To assign an object to a layer

1. Select an object and right-click on it to open its Property Sheet.
2. Click on the Layer property to open a list of existing layers. Select the layer to which you are assigning the selected object.
3. Close the Property Sheet.

Changing a Layer's Visibility at Run Time

At run time, layers can be made visible or invisible by using VBScript to modify the Visible property for "TheView". A button, for example, could execute the following script when clicked:

```
' -----------------------------------------------------
' Sets the Company Layer to invisible
' and the User-Defined Fields Layer to visible.
' -----------------------------------------------------
    TheView.Layer ("Company","False")
```

The syntax for the Layer property is

```
    TheView.Layer (sLayerName, sIsVisible)
```

where sLayerName is a string specifying the name of the layer, and sIsVisible is a boolean ("true" or "false") specifying whether the visible property of the layer is on or off.

Importing Controls

Maximizer Form Designer has the ability to import controls from external dialogs. You can use the Import feature to copy single controls or entire dialogs from Maximizer and other applications simply by pointing to the control or dialog that you wish to copy. Once imported, the controls appear inside your form in edit mode.

When you import controls from another dialog, Maximizer Form Designer imports only the control type and its position relative to the top left corner of the dialog. The control's functionality is not imported from the external dialog.

➤ To import controls from an external dialog

1. Run Maximizer Form Designer.
2. Open the dialog with the controls that you wish to import and try to position it beside the Maximizer Form Designer window, so that you can see the controls. If you cannot position the external dialog where it is visible, use Alt+Tab to toggle between Maximizer Form Designer and the dialog.
3. Select Import Controls from the Edit menu. Your mouse pointer will change to an arrow with a frame on it.
4. Position your mouse pointer over the edge of the control or dialog that you wish to import and click when the desired object appears highlighted. Your mouse pointer will return to a normal arrow.
5. Return to Maximizer Form Designer. The controls that you selected appear in your form.

Locking Your Controls on a Form

To prevent corruption of data and also to prevent your controls from losing their position, you may lock the controls in place. This feature is especially useful if you are sending the form file (*.mxf) to someone else.

➤ To lock your controls on a form

1. Once you have added all the controls on your form, choose Layers from the Layout menu.
2. In the Layers Sheet, select the layer you would like to lock into place.
3. Select the Lock checkbox. This locks all of the objects into the current position in the form.

Creating a Default Form for Use with Form Designer

Maximizer Form Designer can be configured to open a certain form each time you start the program. This feature can be useful if the forms you design have many common features; the default form can then serve as a template for any new forms you create.

➤ **To save a form as a default form**

You cannot save a form as "default.cxf" from within Maximizer Form Designer, as all forms are forced to the file name extension, ".mxf". For this reason, you must rename the form to ".cxf" after it is saved as an ".mxf" file.

1. Open the form that you wish to save as the default form or design the template form if you haven't done so already.
2. Choose Save As from the File menu.
3. Select the folder in which you have installed Maximizer Form Designer. Typically, the program folder for Maximizer Form Designer is the same as your Maximizer program folder (e.g., "\Program Files\Maximizer\").
4. Save the file into the program folder using the file name, "default.mxf".
5. Choose Exit from the File menu. Maximizer Form Designer shuts down.
6. In your Windows Explorer, find the "default.mxf" file, and rename it to "default.cxf".
7. Start Maximizer Form Designer. The form should open automatically.

Populating a Combo Box

You can populate a combo box using one of three methods. The first method involves hard-coding items in a table user-defined field in the Maximizer Form Designer. This method is useful when you would like to add additional items to existing Maximizer fields. For example, if you wished to add the courtesy title "Professor" to the Mr./Mrs. combo box in Maximizer, this method allows you to select the value in Maximizer Form Designer and then to add the additional item.

The second method uses VBScript to create a unique table user-defined field and populate the field with unique values.

The third simply allows you to add a table user-defined field and then assign the values in Maximizer. The instructions for all three methods are explained below.

➤ Populating an Existing Maximizer Combo Box with an Additional Item

1. From the Object bar, select the ComboBox tool.
2. Click inside the form where you would like the combo box placed. Adjust the size to the desired width.
3. Double-click or right-click inside the combo box. The Property Sheet appears.
4. Select the Maximizer field to which you are adding the item. Using the example as explained above, you would select the Mr./Mrs. value.
5. Select the ListItems property. Click the ellipsis button . The List Choices dialog box appears.
6. Click the Add button.
7. In the name field, type the name of the item you are adding. Using the same example, you would type the word "Professor".
8. Click OK. This adds the additional item to the selected table user-defined field. When the form is run, the item will now appear in the available choices for the combo box.

➤ Populating a Combo Box Using VBScript

1. From the Object bar, select the ComboBox tool.
2. Click inside the form where you would like the combo box placed. Adjust the size to the desired width.
3. Double-click or right-click inside the combo box. The Property Sheet appears.
4. Click the ellipsis button located on the Events tab, next to the EventClick property. The VBS Mini-Editor appears.
5. Insert your script. To access the available properties in Run mode, you must type the name of your label followed by a period. (.) This activates the VBS Mini-Editor.
6. Click OK to close the editor and then click the close button to close the Property Sheet. This makes your changes to the form. When you run the form or attempt to close it, you are prompted to save your changes.

Example of Populating a Combo Box

The following example demonstrates a simple method for populating a combo box with field items using VBScript. In this example, the combobox "cboMr_Ms" is being populated with list items.

```
' ---------------------------------------
' Add items to combo box
' ---------------------------------------
    cboMr_Ms.AddString "Mr."  ' Add comboBox item option
    cboMr_Ms.AddString "Sir."  ' Add comboBox item option
    cboMr_Ms.AddString "Mrs." ' Add comboBox item option
    cboMr_Ms.AddString "Miss." ' Add comboBox item option
```

The combo box can be set to display a certain item in the list, as in the following example:

```
' ---------------------------------------------------------
' Retrieves the currently selected Combo box item.
' ---------------------------------------------------------
    lCurrentSelected = cboMr_Ms.Selection ' Returns the
currently selected list index value. List starts with 0
```

Example of Populating a Combo Box with User-Defined Field Items

The following example illustrates one method for populating a combo box using VBScript.

This sample code is found in the EventChange for a combo box which lists the UDF items. This code takes the selected item and appends it the current selected items in the UDF table edit control.

```
' ---------------------------------------------------------
' Populate ComboBox with Items from UDF
' * The GetListOfUdfTableItems returns a string
' in which items are separated by Chr$(13) + Chr$(10) and
' not ",". Therefore, separate the return string
' into it's items by searching for Chr$(13) & Chr$(10)
' ---------------------------------------------------------
Dim iCommaPos As Integer '
Dim iItemLen As Integer
Dim sItems As String
Dim objMaxApp As Object ' Define Application Object variable
Dim objMaxAttach As Object
Dim objMaxRec As Object
```

```
Dim iStart As Integer



Set objMaxAttach =
CreateObject("Maximizer.AttachToCurrentInstance")
Set objMaxApp = objMaxAttach.GetApplicationObject  ' Set
objMaxApp to Current Maximizer Application
Set objMaxRec = objMaxAttach.GetCurrentRecordObject


sItems = objMaxApp.GetListOfUdfTableItems("Product Type")
' ----------------------------------------------------
' Parse as UDF Table of it's items...
' ----------------------------------------------------
iItemLen = Len(sItems)' Set initial values
iStart = 1     ' set initial values
iCommaPos = 1' set initial values


Do While iCommaPos < iItemLen And iCommaPos <> 0
  iCommaPos = InStr(iStart, sItems, Chr$(13) & Chr$(10),
vbBinaryCompare)


  If iCommaPos = 0 Then Exit Do ' No match, end of string


  cboType.AddString Mid$(sItems, iStart, iCommaPos -
(iStart))
  iStart = iCommaPos + 2
Loop


Set objMaxRec = Nothing' Release Object
Set objMaxApp = Nothing ' Release object
Set objMaxAttach = Nothing  ' Release Object
```

➤ Populating A Combo Box In Maximizer

1. Add the user-defined field in Maximizer Form Designer. Adding User-defined fields to Your Forms. Make sure that you select Table as the type.
2. In Maximizer, select Setup User-Defined Fields from the File menu.
3. In the Setup User-Defined Fields dialog box, select the table user-defined field to which you are adding an item.

4. Click the Items button.

   – or –

   Double-click the selected user-defined field. The Setup Items dialog box appears.

5. In the Setup Items dialog box, click the Add button.

6. In the Add Item dialog box, type an item description. You may later choose this item as a value for a field in an Address Book entry.

7. When you are finished, click OK. Repeat steps 6 to 8 for each item you are adding.

8. Click the Close button, and then click Close again.

Using Custom Forms in Maximizer

Using a custom form in Maximizer can be as easy as inserting an entry and choosing the custom form to use. In some cases, however, you may have to register the file Viewer.dll or make a change to the security before you can use your form in Maximizer.

Registering the Maximizer Form Designer Viewer.dll

In order to use a form you have created with the Maximizer Form Designer, the Viewer.dll must be registered on your computer. This registration should automatically take place the first time you run the Maximizer Form Designer; however, if it does not, use the following procedure to do so:

1. Click the Start button, then choose Run.
2. Type **regsvr32** followed by the folder path where you have Maximizer installed. For example, **"c:\progra~1\maximi~1\viewer.dll"**. Make sure you follow the 8-character naming convention; if your folder names are longer than eight characters, type the first 6 followed by a ~1.
3. Click OK. You are prompted with a message informing you if the Viewer.dll has been successfully registered on your computer.

Configuring Security in Maximizer

There are a few steps you must follow to set up security for use of a form you have created in Maximizer Form Designer with Maximizer.

➤ **Setting up security to allow the use of an alternate form in Maximizer**

1. In Maximizer, select File > Preferences. The Security tab should be active—if not, click the Security tab.
2. Click the System Defaults button. the System Defaults dialog box appears.
3. De-select the Disable Creation of Form UDFs checkbox. A *UseAlternateForm type of user-defined field is created in Maximizer. The *UseAlternateForm user-defined field allows you use dialog boxes (forms), other than the default dialog boxes (forms) provided with Maximizer.
4. Click OK on the System Defaults dialog box.
5. Click Apply and then click OK, on the Security tab.

Depending on what type of entry you have associated with the form—Company, Individual or Contact—the next time you create that type of entry, you will be prompted to select the name of the alternate form you would like to use. Follow the instructions in the topic Creating an Entry in Maximizer Using an Alternate Form.

Using an Alternate Form to Create a New Address Book Entry

➤ **To create an entry in Maximizer Using an Alternate Form**

This procedure works the same way when you are modifying an entry that was created using an alternate form.

1. After you have designed your form, choose Save from the File menu in Maximizer Form Designer.
2. Name the form file. You should choose a meaningful name so its purpose is easily determined, since you may want to have more than one alternate form in Maximizer.
3. Specify the type of entry you would like the form associated with in the Change Form Type dialog box—Individual, Company or Contact, and then click OK. You can modify this information at any time by selecting File > Change Form Type in Maximizer Form Designer.
4. In Maximizer, click the New button on the Standard toolbar or select Edit > Add and then choose the type of entry to which you have associated the form. For instance, if you have chosen Company as the type of entry associated with the form, choose New/Add > Company. The Select Form dialog box appears displaying a list of all the forms available for use. This list includes forms for all types of entries—Companies, Individuals and Contacts.
5. Select the form you would like to use and click OK. This is where meaningful names are useful in that if you have more than one form listed, you may easily determine what the appropriate choice is. The chosen form appears.
6. Enter your information in the form. This creates the entry in Maximizer.

Tutorials

By default, Maximizer Form Designer opens with a default form template ready for you to design. At any time, you can select File > New to begin designing a new form.

Designing a form is essentially a visual process, allowing you to achieve impressive results in a very short period of time. Form Designer provides several form objects, including text labels, bitmaps, edit controls, list boxes and spinners, among others.

This section contains two tutorials which guide you through the steps of creating simple forms. Tutorial 1 illustrates how to design a simple custom form, and Tutorial 2 illustrates how to design a “tabbed” form using layers, rather than a tab control.

Tutorial 1—Creating a Custom Form

In this tutorial, you will create a simple custom form with an edit box, a label for the edit box, and an OK button.

➤ To create a custom form

1. Start Maximizer Form Designer. You should see the default template in the Form Designer workspace. If not, select File > New from the main menu. Alternatively, you can press Ctrl+N to create a new blank form.
2. To add controls, the Object bar must be visible. Select View > Object bar or press F2 to display the Object bar if it is not visible.
3. To add a label to the form, select the Text tool from the Object bar. Then, use your mouse to position and size the object. To do so, click in the area where you would like the upper left corner of the object placed and drag the mouse to the desired size of the control. Select the default text in the object and replace it with "&Company:" (the ampersand is required to provide an accelerator key).
4. Right-click inside the text box to activate the Property Sheet. Specify the attributes for the label.
5. To add an input field, select one of the tools from the Object bar—such as the EditBox tool—and position the object on the form to the right of the text label. Again, use your mouse to position and size the object.
6. Right-click inside the edit box to bring up the Properties Sheet. Attributes are listed on the Normal tab and Events on the Events tab. On the Normal tab, scroll down to Maximizer Field and click inside the column on the right. All of the available fields

(including user-defined fields) are shown in the drop down listbox. Selecting Company from the list will cause the editbox to automatically be attached to the Company field in the current Maximizer Address Book folder.

7. To add a button, select the Button tool from the Object bar and place it on the form. Available Button Types include OK and Cancel buttons and their respective actions. Choose the OK item as the Button Type. You may also perform an event click by choosing this as the Button type and then specifying the action in the VBS Mini-Editor.

8. Select Edit > Set Tab Order to change the tab order of the controls.

9. Select File > Change Form Type and set the type to Company. If you do not set the Form Type now, you are prompted to do so when you save the form.

10. Select File > Save As and save the form as "Tutorial1.mxf".

11. Select Layout > Edit, click the Edit button or press Ctrl+E to test the form.

12. You now have a working form! Open Maximizer, make the Address Book the active window and press Insert.

13. Select "Tutorial1" from the Select Form dialog. Enter a Company name and press OK.

Your new entry appears in the Address Book.

Tutorial 2—Creating a Tabbed Form Using Layers

There are two different methods you may use to create a tabbed form with Maximizer Form Designer. The first method involves using a tab control and creating a number of forms—one for each tab on the form. The second method uses a feature known as Layers in Maximizer Form Designer.

In this tutorial we'll use layers, rather than a tab control, to create a tabbed form.

➤ Step 1—Create the Default Layer

1. Start Maximizer Form Designer. You should see a new blank form in the workspace.

2. Select the Button object from the Object bar and place the button in the top left corner of your form.

3. Right-click on this button to open its Property Sheet. Make sure the box at the top of the Property Sheet says “Button1”. (If it

says “TheFrame/TheView”, click on your new button, then right-click on it again to open the Property Sheet for the button, not the Frame.)

4. Set the ButtonShape property in the Property Sheet to “2 - Property Tab (Active)”.
5. Change the Text property from “Button1” to “Company”.
6. Select the Button object from the Object bar and place the button just to right of the first button—labeled “Company”. Your new button should be labeled “Button2”.
7. Right-click on the second button to open its Property Sheet.
8. Set the ButtonShape property in the Property Sheet to “1 - Property Tab (Inactive)”.
9. Change the Text property from “Button2” to “Phone”.
10. Select the Button object from the Object bar and place the button at the bottom right corner of your form.
11. Right-click on this new button to open its Property Sheet.
12. Set the ButtonType property to “4 - OK”.
13. Set the Text property to “&OK” (the ampersand assigns an accelerator key to the letter “O”).
14. You should now have three buttons on your form: “Company” and “Phone” in the top left corner and “OK” in the bottom right corner. You are now ready to create the next layer.

➤ Step 2—Create Layer1

1. Select Layers from the Layout menu. The Layers Sheet opens.
2. Click Add to create a new layer—“Layer1”—and click OK to close the Layers Sheet.
3. Select the Text object from the Object bar, and place it near the left edge of your form. The new text label says “Text2”. Right-click on this text label to open its Property Sheet.
4. Set the Alignment property to “3 - Right”.
5. Change the Text property to “Company :”.
6. Make sure the Layer property is set to “Layer1”.
7. Select the EditBox object from the Object bar and place it just to the right of the Text object, which should say “Company :”. Right-click on the EditBox to open its Property Sheet.
8. Set the Maximizer Field property to “Company”.

9 Make sure the Layer property is set to "Layer1".

10 You are now done creating Layer1. Your form now has the three buttons on it, as well as an EditBox labeled "Company :". You are now ready to create the next layer.

➤ Step 3—Create Layer2

1. Select Layout from the Layers menu. The Layers Sheet opens.
2. Select “Layer1” in the list of layers and uncheck the View box. The “eye” icon disappears from beside “Layer1” and your Company edit box becomes invisible on the form.
3. Click Add to create a new layer. Make sure the new layer is named “Layer2” and click OK to close the Layers Sheet.
4. Select the Text object from the Object bar and place the Text object near the left edge of the form. Right-click on the new Text object to open its Property Sheet.
5. Set the Alignment property to “3 - Right”.
6. Change the Text property to “Phone :”.
7. Make sure the Layer property is set to “Layer2”.
8. Select the EditBox object from the Object bar and place it just to the right of the Text object (which should say “Phone :”). Right-click on this new EditBox to open its Property Sheet.
9. Set the Maximizer Field property to “Phone1”.
10. Make sure the Layer property is set to “Layer2”.
11. You have now finished Layer2. Your form should now have three buttons and an EditBox labeled “Phone :” (the “Company :” edit box is still invisible). You are now ready to add scripting to your form.

➤ Step 4—Add Scripting to Your Form

1. Select Layers from the Layout menu. The Layers Sheet will open.
2. Select “Layer2” in the list of layers and uncheck the View checkbox.
3. Select “Layer1” in the list of layers and check the View checkbox. The “Company” edit box should again be visible in your form. Click OK to close the Layers Sheet.
4. Select the “Company” button in your form, then right-click on it to open its Property Sheet. Select the Events tab.
5. Click the word “(None)” beside “EventClick” to open the VBS Mini-Editor. Enter the following text into the editor exactly as it appears here:

```
TheView.Layer ("Layer2","False")
TheView.Layer ("Layer1","True")
```

```
This.ButtonShape 2
Button2.ButtonShape 1
```

6. Click OK when you are done typing the script.
7. Select the "Phone" button in your form, then right-click on it to open its Property Sheet. Select the Events tab.
8. Click the word "(None)" beside "EventClick" to open the VBS Mini-Editor. Enter the following text into the editor exactly as it appears here:

```
TheView.Layer ("Layer1","False")
TheView.Layer ("Layer2","True")
This.ButtonShape 2
Button1.ButtonShape 1
```

9. Click OK when you are done typing the script.
10. Select Save As from the File menu. The Save As dialog opens. Save your form as "Tutorial2.mxf".
11. Select Edit from the Layout menu, click the Edit button or press Ctrl+E to test the form.
12. You now have a working form! Open Maximizer, make the Address Book the active window and press Insert.
13. Select "Tutorial2" from the Select Form dialog. Enter a Company name, click the Phone tab, enter a phone number, and click OK.

Your new entry appears in the Address Book.

CHAPTER 3

Objects

Draw Custom Forms Quickly with Objects

In this chapter...

Objects Available in Maximizer Form Designer

The design objects allow you to create powerful forms to represent the data stored in your Maximizer Address Book folder. You can use the objects only within the Maximizer environment.

Maximizer Form Designer includes 22 objects, which are available from the Object bar. The built-in objects are as follows:


ActiveX


AnimCtrl


Bitmap


Button


CheckButton


ComboBox


DateTimePicker


EditBox


EditDoubleBox


Frame


Line


ListBox


ListCtrl


EditBox


MultiEditBox


RadioButton


Slider


Spinner


TabCtrl


Text


TextVar


TreeCtrl

Frame/View

Frame/View is the term used to describe the workspace in which forms are designed. Although strictly speaking Frame/View is not an object, it does have many of the same properties, methods, and events as the objects.

In design mode, you can right-click on the Frame/View to bring up its property sheet, in which you can modify properties such as the Title and Background color, enable or disable scrollbars, etc.

Frame/View Properties

AutoRecord
BorderStyle
Color
CursorPointer
Font
Height
HelpContextID
LocalDecls
LocalVariables
MaximizeBox
MinimizeBox
PrintScale
ReportLock
ScrollBars
ScrollHeight
ScrollWidth
Title
ToolTipText
ViewLayers
WhatsThisHelp
Width

Frame/View Methods

ClearDataAll
GUIEventAll
OnUpdate
OnUpdateResize
PumpDataAll
Cancel
FormFind
IsFormOpen

OpenDoc
PumpData
Record
RunClick
RunInitialize
RunTerminate

ActiveX

ActiveX controls, previously called custom or OLE controls, are custom controls created by third party to accomplish a specific task.

ActiveX Properties

BackColor
Bottom
CursorPointer
Enable
Group
HelpContextID
Layer
Left
Right
Tabstop
ToolTipText
Top
Visible

ActiveX Methods

GetDlgCtrlID
LayerName
OnUpdate
OnUpdateResize
OnUpdateStyle
RunInitialize
RunPumpData
SetFocus

AnimCtrl

The AnimCtrl object (or "Animation Control") allows the user to easily toggle between images at run time.

AnimCtrl Properties

AutoPlay
BorderDrawn
Bottom
CursorPointer
Enable
FilePath
Group
HelpContextID
Layer
Left
Right
Tabstop
ToolTipText
Top
Visible

AnimCtrl Methods

GetDlgCtrlID
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
Play
RunChange
RunInitialize
RunPumpData
SetFocus
Stop

Bitmap

The Bitmap object contains "picture" information like bitmaps and JPEG images. This control manages all aspects of storing, retrieving and displaying the images without the need for any knowledge of the format or any special programming to display these images.

The uses of the Bitmap object include any application that requires that images are stored and displayed, such as product catalogues, contact management, real-estate applications and so on.

Bitmap Properties

AnchorSnaps

Bitmap

Bottom

CursorPointer

Enable

Font

ForeColor

HelpContextID

Layer

Layout

Left

Right

ToolTipText

Top

Visible

Bitmap Methods

LayerName

Move

OnUpdate

OnUpdateResize

RunClick

RunInitialize

RunPumpData

Button

The Button object enables you to place on your form a push button with a pre-defined action. The Button object's ButtonType property allows you to select one of six common button functions: Cancel, EventClick, Goto, Help, OK and Record. In addition to these pre-defined actions, you can add scripting to a button in order to customize its functionality.

Button Properties

BackColor
Bitmap
Bottom
ButtonShape
ButtonType
CursorPointer
Default
Enable
Font
ForeColor
GotoPath
Group
HelpContextID
Layer
Layout
Left
Right
Tabstop
Text
ToolTipText
Top
Visible

Button Methods

GetDlgCtrlID
LayerName
Move

OnUpdate

OnUpdateResize

OnUpdateStyle

RunClick

RunInitialize

RunPumpData

SetFocus

CheckButton

The CheckButton object is a standard CheckButton control. As such, it normally has two possible states: 'checked' and 'unchecked', although it is possible to configure the object's TriState property to enable a third state, which of course is 'neither'.

Other properties relate to the adjoining text label, the mouse cursor when the mouse is over the control, the tooltip text when the mouse is over the control, and so on.

CheckButton Properties

AlignTextLeft

Backcolor

Bottom

CursorPointer

Enable

Font

ForeColor

Group

HelpContextID

HelpHotButton

Layer

Left

Maximizer Field

Right

Tabstop

Text

ToolTipText

Top

TriState

Visible

CheckButton Methods

GetDlgCtrlID
IncrValue
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus

ComboBox

The ComboBox object is a standard ComboBox control. A ComboBox differs from a list box in that it allows the user to enter text into the field, as well as selecting from the list activated by the drop-down arrow.

ComboBox Properties

Accelerator
BackColor
Bottom
ComboType
CursorPointer
Enable
Font
ForeColor
Group
HelpContextID
HelpHotButton
Layer
Left
ListItems

Maximizer Field
NumDropped
Right
Sort
Tabstop
ToolTipText
Top
UseColors
Visible

ComboBox Methods

AddString
DeleteString
GetDlgCtrlID
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus

DateTimePicker

The DateTimePicker is like two objects in one, enabling you to place both date and time fields on a form. The DateTimePicker's UpDown property lets you choose either up and down "spinner" type buttons, which increment and decrement the displayed date/time, or a down arrow button, which pops up a calendar for selecting a date.

Using a DTP Format String

Maximizer cannot store the time along with a bound date field. Only the date will be stored. To store the time, you will need to write additional script code to get the date/time. The format string for the DTP can be set and retrieved under script control. The properties that are useful for doing this are Date and Time.

The DateTimePicker (DTP) control relies on a format string to determine how it will display fields of information, and provide greater flexibility for your application. The format characters that comprise the format string define the DTP control's display and field layout.

You can add body text to the format string. For example, if you want the control to display the current date with the format, "Today is: 04:22:31 Thursday Oct 15, 1998", the format string would be, "'Today is: 'hh':'m':'s ddddMMMdd', 'yyy".

Notice that segments of nonformat characters in the preceding example are delimited by single quotation marks. Failure to surround body text in this way will result in unpredictable display by the DTP.

Valid DTP Format Characters

Date and Time Picker supports the following format characters:

"d"	The one- or two-digit day
"dd"	The two-digit day
"ddd"	The three-character weekday abbreviation
"dddd"	The full weekday name
"h"	The one- or two-digit hour in 12-hour format
"hh"	The Two-digit hour in 12-hour format
"H"	The one- or two-digit hour in 24-hour format
"HH"	The two-digit hour in 24-hour format
"m"	The one- or two-digit minute
"mm"	The two-digit minute
"M"	The one- or two-digit month number
"MM"	The two-digit month number
"MMM"	The three-character month abbreviation
"MMMM"	The full month name
"t"	The one-letter AM/PM abbreviation (i.e., AM is displayed as "A")
"tt"	The two-letter AM/PM abbreviation
"x"	The callback field. The control still uses the other valid format characters, and queries the owner to fill in the "X" portion. The owner must be prepared to handle the DTN_WMKEYDOWN, DTN_FORMAT, and DTN_FORMATQUERY notification messages. Multiple "X" characters can be used in series to signify unique callback fields.
"y"	The one-digit year (i.e., "1998" would be displayed as "8")

"yy"	The last two digits of the year
"yyy"	All four digits of the year

DateTimePicker Properties

AllowNoSetTime
BackColor
BetweenMonthsBackColor
Bottom
CalendarBackColor
CursorPointer
Enable
Font
ForeColor
Format
Group
HelpContextID
Layer
Left
Maximizer Field
NonCurDatesTextColor
Right
Tabstop
TextColor
TitleBackColor
TitleTextColor
ToolTipText
Top
UpDown
Visible

DateTimePicker Methods

GetCtrlID
GetWindowText
LayerName
Move
OnUpdate

OnUpdateResize
OnUpdateStyle
RunInitialize
RunPumpData
SetFocus
SetWindowText

EditBox

The EditBox object is a standard single line text control for display or entry of a single line of alphanumeric text.

EditBox Properties

Accelerator
Alignment
BackColor
BorderDrawn
Bottom
CaseOrPassword
CursorPointer
Enable
Font
ForeColor
Group
HelpContextID
HelpHotButton
Layer
Left
Maximizer Field
ReadOnly
Right
Tabstop
Text limit
ToolTipText
Top
UseColors

Visible

EditBox Methods

GetCtrlID
GetWindowText
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus
SetWindowText

EditDoubleBox

The EditDoubleBox object is a standard text box for numeric data. The EditDoubleBox provides the user additional control over edit contents with exposed properties.

EditDoubleBox

Accelerator
Alignment
BackColor
BorderDrawn
Bottom
CaseOrPassword
CursorPointer
Enable
Font
ForeColor
Group
HelpContextID
HelpHotButton
Layer

Left
Maximizer field
MinimumEq
MaximumEq
ReadOnly
Right
Tabstop
Text limit
ToolTipText
Top
UseColors
Visible

EditDoubleBox Methods

GetCtrlID
GetWindowText
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus
SetWindowText

Frame

The Frame object is a graphical object used to visually group two or more controls in a form, or separate controls into groups of similar functions.

Frame Properties

Alignment
BackColor
BorderColor

BorderStyle
BorderWidth
Bottom
CursorPointer
Enable
Font
Forecolor
HelpContextID
HiliteColor
Layer
Left
Right
Text
ToolTipText
Top
Visible

Frame Methods

LayerName
Move
OnUpdate
OnUpdateResize
RunInitialize
RunPumpData

Line

The Line object is simply a graphical element for use on forms. You have control over color, thickness, length, and the presence or absence of arrowheads.

Line Properties

Arrowhead
ArrowheadHeight
Bottom
CursorPointer
Enable

ForeColor
HelpContextID
Layer
Left
PenStyle
Right
RotateAngle
ToolTipText
Top
Visible
Width

Line Methods

LayerName
Move
OnUpdate
OnUpdateResize
RunClick
RunInitalize
RunPumpData

ListBox

The ListBox object is a standard list box control. A ListBox simply presents the user with a series of choices in a drop-down list—if the user needs to be able to enter a value not in the list, a ComboBox should be used instead.

ListBox Properties

Accelerator
BackColor
Bottom
CursorPointer
DoLBN_CHANGE
Enable
Font
ForeColor

Group
HelpContextID
HorizontalScroll
Layer
Left
ListID
ListItems
Maximizer Field
RemoveSelection
Right
Sort
Tabstop
ToolTipText
Top
UseColors
UseTabstops
VerticalScroll
Visible
WantKeyInput

ListBox Methods

AddString
DeleteString
GetCtrlID
GetText
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
ResetContent
RunChange
RunInitialize
RunPumpData
SetFocus
Value

ListCtrl

The ListCtrl object (or "List Control") displays a list of items from which the user can select one or more. If the number of items exceeds the number that can be displayed, a scroll bar is automatically added to the ListBox control. This particular list control can also be referred to as a grid control, due to it's column display.

ListCtrl Properties

Bkcolor

Bottom

CursorPOinter

Enable

Group

HelpContextID

Layer

Left

Right

Tabstop

TextBkColor

TextColor

ToolTipText

Top

Visible

ListCtrl Methods

DeleteAllItems

DeleteColumn

DeleteItem

FindItem

GetBkColor

GetCountPerPage

GetDlgCtrlID

GetItemCount

GetItemText

GetSelectedCount

GetSelectedItem

GetTextBkColor

GetTextcolor

InsertColumn

InsertItem

LayerName

Move

OnUpdate

OnUpdateResize

OnUpdateStyle

RunChange

RunInitialize

RunPumpData

SetBkColor

SetFocus

SetItem

SetItemCount

SetItemText

SetTextBkColor

SetTextColor

MaskedEditControl

The MaskedEditControl is designed to restrict data entry to ensure data conforms to a specific format. Uses of this object include storage of pre-formatted telephone numbers, credit card numbers, US social security numbers, driver's licence numbers and so on.

Character Masks

A character mask is a designer supplied mask specifying which characters are allowed to be placed in the following character mask subgroup.

It has the syntax:

```
[ (start character)(end character),(single character)...]
eg. [bf,DM,X,\,]
```

meaning the next character mask subgroup will accept the characters:

'b' through to 'f',

'D' through to 'M',

the letter 'X',

the letter ','

(note the use of the literal character '\' to specify that the character ',' is not to be taken as a control character)

It is invalid to have a right justified subgroup and then a left justified subgroup without any literals separating the groups (otherwise there is contention to which subgroup gets control of the cursor and input).

This character mask will be used for the next character mask subgroup (ie 'M' or 'm').

A mask will need to be specified for each character mask subgroup (NOT for every character mask character).

eg. '[aZ, 1,5,/]]RMMMMMMM'

This will allow any alphabetic characters (a-Z), the numbers '1' and '5' and the symbol ']' to be input into the right justified 7 character mask characters.

Mask Character	Description
#	Digit placeholder
9	Optional digit placeholder
A	Alphanumeric placeholder
a	Optional alphanumeric placeholder
&	Printable character placeholder
C	Optional printable character placeholder
?	Letter placeholder
z	Optional letter placeholder
[	start of character mask specification
]	End of character mask specification
M	Character mask placeholder
m	Optional character mask placeholder
.	Decimal Character—the actual character used is the one specified in your international settings
,	Thousands separator—the actual character used is the one specified in your international settings
:	Time Separator—the actual character used is the one specified in your international settings
/	Date Separator—the actual Character used is the one specified in your international settings
$	Currency Symbol—the actual Character used is the one specified in your international settings
\	The character following is a literal, which is used to specify literal characters that are used as mask control characters (ie #,9,A,a ... etc.)
>	Start of UpperCase conversion—converts all following characters to uppercase (until an end case or lower case conversion)

Mask Character	Description
<	Start of LowerCase conversion—converts all following characters to lowercase (until an end case or upper case conversion)
|	End of Case conversion—stops converting case
L	The next subgroup will be left justified
R	The next subgroup will be right justified
Literals	All other characters are displayed as literals

MaskedEditControl Properties

Properties
Accelerator
BackColor
BorderDrawn
Bottom
CaseOrPassword
CursorPointer
Enable
Font
ForeColor
Group
HelpContextID
HelpHotButton
Layer
Left
Mask
Maximizer Field
ReadOnly
Right
Tabstop
ToolTipText
Top
UseColors
Visible

MaskedEditControl Methods

GetCtrlID

GetWindowText
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus
SetWindowText

MultiEditBox

The MultiEditBox object is a standard multi-line text control with optional scroll bars.

MultiEditBox Properties

Accelerator
Alignment
BackColor
BorderDrawn
Bottom
CaseOrPassword
CursorPointer
Enable
Font
ForeColor
Group
HelpContextID
HelpHotButton
HorizontalScroll
Layer
Left
Maximizer Field
ReadOnly

Right
Tabstop
Text limit
ToolTipText
Top
UseColors
VerticalScroll
Visible

MultiEditBox Methods

GetCtrlID
GetWindowText
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus
SetWindowText

RadioButton

The RadioButton object is designed to provide typical radio button functionality on a form. It also provides data awareness, with the ability to assign any value to a radio button. This enables the use of radio buttons to set values such as 'Small', 'Medium' and 'Large' to a field in the Address Book folder for example.

RadioButton Properties

AlignTextLeft
BackColor
Bottom
CursorPointer
Enable

Font
ForeColor
Group
HelpContextID
Layer
Left
ListItems
Maximizer Field
Right
Tabstop
ToolTipText
Top
Visible

RadioButton Methods

GetDlgCtrlID
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus

Slider

A Slider control contains a slider and optional tick marks. You can move the slider by dragging it, clicking the mouse to either side of the slider, or using the keyboard.

Slider Properties

AutoRecord
BorderStyle
Color
CursorPointer

Font
Height
HelpContextID
LocalDecls
LocalVariables
MaximizeBox
MinimizeBox
PrintScale
ReportLock
ScrollBars
ScrollHeight
ScrollWidth
Title
ToolTipText
ViewLayers
WhatsThisHelp
Width

Slider Methods

GetCtrlID
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus

Spinner

A Spinner control has a pair of up/down arrow buttons which the user can click to increment or decrement a value, such as a scroll position or a value in an associated control (i.e., a "buddy" control).

Spinner Properties

Accelerator
Bottom
CursorPointer
DecimalBase
EditBuddy
Enable
Group
HelpContextID
IncrAccelerator
layer
Left
Maximizer Field
MaximumEq
MinimumEq
Orientation
Right
Tabstop
ToolTipText
Top
Visible
WrapAround

Spinner Methods

GetCtrlID
LayerName
Move
OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus

TabCtrl

A TabCtrl object (or "tab control") allows you to create two or more tabbed pages of controls in your form. This control can be especially useful if you need to create a form with many controls and objects, but there isn't enough space available in the standard form.

When you create a form that has a tab control on it, only the first page of the form is stored with the tab. When the user clicks subsequent tabs, the tab form will be loaded from a folder named "TabForms" under the folder in which the form resides. For example, if you design a form called "Hospital" with three tab pages, the following files are required:

<Maximizer path>\Forms\Hospital.mxf>

<Maximizer path>\Forms\TabForms\Hospital1.mxf>

<Maximizer path>\Forms\TabForms\Hospital2.mxf>

Tab Control Properties

Bottom

ButtonShape

CursorPointer

Enable

Font

Group

HelpContextID

Layer

Left

ListItems

Right

Tabstop

ToolTipText

Top

Visible

Tab Control Methods

GetCtrlID

LayerName

Move

OnUpdate
OnUpdateResize
OnUpdateStyle
RunChange
RunInitialize
RunPumpData
SetFocus

Text

The Text object simply displays static text. This object can be used as a label for other objects, similar to the Frame object.

Text Properties

Alignment
AnchorSnaps
BackColor
BorderStyle
Bottom
CursorPointer
Enable
Font
ForeColor
HatchStyle
HelpContextID
Hilitecolor
Layer
Left
Right
ShadowStyle
Text
ToolTipText
Top
Visible
Width

Text Methods

LayerName
Move
OnUpdate
OnUpdateResize
RunClick
RunInitialize
RunPumpData

TextVar

The TextVar object is a dynamic text object which, like the Text object, may be used as a label for other objects. TextVar can have variables attached to it using VBScript in order to display dynamic text as a label or message.

TextVar Properties

Alignment
AnchorSnaps
Backcolor
BorderStyle
Bottom
CursorPointer
Enable
Font
ForeColor
Format
HatchStyle
HelpContextID
HiliteColor
Layer
Left
Right
ShadowStyle
ToolTipText
Top
ValueEq

Visible

Width

TextVar Methods

LayerName
OnUpdate
OnUpdateResize
RunClick
RunInitialize
RunPumpData

TreeCtrl

The TreeCtrl object (or "Tree Control") is typically used to display the headings in a document, the entries in an index, the files and directories on a disk, or any other kind of information that might usefully be displayed as a hierarchy.

TreeCtrl Properties

Bottom
CursorPointer
Enable
Group
HelpContextID
Layer
Left
Right
Tabstop
ToolTipText
Top
Visible

TreeCtrl Methods

DeleteAllItems
DeleteItem
Expand
GetChildItem
GetCount
GetCtrlID

GetFirstVisbleItem

GetIndent

GetItemText

GetNextItem

GetNextSiblingItem

GetNextVisibleItem

GetParentItem

GetPrevSiblingItem

GetNextSiblingItem

GetRootItem

GetSelectedItem

InsertItem

ItemHasChildren

LayerName

Move

OnUpdate

OnUpdateResize

OnUpdateStyle

RunChange

RunInitialize

RunPumpData

Select

SelectItem

SelectSetFirstVisible

SetFocus

SetIndent

SetItemText

CHAPTER 4

Methods

Add Functionality to Form Designer Objects

In this chapter...

Common Methods

The following methods are common to all Maximizer Form Designer Objects.

GetDlgCtrlID () As Long

Allows Enable Script to retrieve the dialog control ID.

GetWindowText () As String

Allows Enable Script to retrieve the text from the called Object.

Move (ByVal P1 As Long, ByVal P2 As Long)

Moves an Object according to the inputted arguments.

```
Object.Move P1, P2
```

Object	Required. The Object that is to be moved.
P1	Required. Single-precision value indicating the horizontal coordinate (x-axis) for the left edge of Object.
P2	Required. Single-precision value indicating the vertical coordinate (y-axis) for the top edge of Object.

OnUpdate ()

Allows you to perform an action upon update of the contents of a control.

OnUpdateResize ()

Allows you to perform an action upon the resizing of a control.

OnUpdateStyle ()

Allows you to perform an action upon the update of a control's style.

SetFocus ()

Moves the focus to the specified Object.

```
object.SetFocus
```

You can only move the focus to a visible form or control. You also can't move the focus to an Object if the Enabled property is set to False. If the Enabled property has been set to False at Design Time, you must first set it to True before it can receive the focus using the SetFocus method.

SetWindowText (ByVal P1 As String)

Allows Enable Script to set text back to the Object that is being called.

Text Box Method Examples

The following examples contain sample script in which a text box method is used.

GetWindowText() as String

```
' ----------------------------------------------------------
' Examples the GetWindowText method
' which is available with Edit controls
' txtLast is the name of an edit control.
' ----------------------------------------------------------
If "" = txtLast.GetWindowText Then ' If blank, force user to
insert text.
    Beep
    txtLast.SetFocus ' SetFocus to txtLast control
End If
```

SetWindowText(ByVal sValue as String)

```
' ----------------------------------------------------------
' Examples use of GetWindowText and
' SetWindowText method for edit control
' object, txtLast is a edit control object.
' ----------------------------------------------------------

sTempStr = txtLast.GetWindowText ' Retrieves the value of
txtLast control
    ' Capitalize the first Character of txtLast text.
sTempStr = UCase(Mid$(sTempStr,1,1)) & Mid$(sTempStr,2,
Len(sTempStr) -1)
```

```
    ' Writes the change back to txtLast edit control.
txtLast.SetWindowText( sTempStr)
BackColor(lColor as Long)
' ----------------------------------------------
' Examples use of backColor property
' ----------------------------------------------
txtLast.BackColor &H0000FFFF& ' Sets the backColor to Yellow
```

ForeColor(lColor as Long)

```
' ----------------------------------------------
' Examples use of ForeColor property
' ----------------------------------------------
txtLast.ForeColor &HFF000000& ' Sets the backColor to Red
```

Enable(iMode as integer)

```
' ----------------------------------------------
' txtRec is a text box control object, stop
' users from editing control by making
' Enable equal False.
' ----------------------------------------------
txtRecType.Enable 0 ' Disable control.
```

Using the Select Case Statement

```
Dim oMaxRec As Object
Dim oMaxAttach As Object
Dim sRecType As String

Set oMaxAttach =
CreateObject("Maximizer.AttachToCurrentInstance")
Set oMaxRec = oMaxAttach.GetCurrentRecordObject ' Get OLE
CurrentRecord Object

' ----------------------------------------------------------
' Define the Text of the txtRecType text
' box based on the Maximizer record type.
' ----------------------------------------------------------
sRecType = oMaxRec.GetFieldValue
Select Case sRecType
```

```
    Case "1":
        txtRecType.SetWindowText ("Company")
    Case "2":
        txtRecType.SetWindowText("Individual")
    Case "31":
        txtRecType.SetWindowText("Company Contact")
End Select
    txtRecType.Enable 0 ' Disable control.


Set oMaxRec = Nothing
Set oMaxAttach = Nothing
```

Attaching to CurrentInstance and OLE Automation Application Object

```
Dim oMaxAttach = Object
Dim oMaxApp = Object

Set oMaxAttach =
CreateObject("Maximizer.AttachToCurrentInstance")
Set oMaxApp = oMaxAttach.GetApplicationObject  ' Set oMaxApp
to Current Maximizer Application

oMaxApp.ActiveWindow("HotList")
If vbYes = MsgBox ("Return back to Client
Window?",vbYesNo,"Form Designer.") Then
    oMaxApp.ActiveWindow("Client")
End If

Set oMaxApp = Nothing ' Release object.
Set oMaxAttach = Nothing  ' Release Object.
```

Other Methods

AddHighlightDate(DATE Date)

The AddHighlightDate(DATE Date) method adds Date to the list of dates to be shown as highlighted dates (this feature is used by the forthcoming Time Manager product to show dates that have an appointment).

```
void    AddHighlightDate(DATE Date)
```

AddString

The AddString method adds a string to the end of the list in the list box of a combo box or at the sorted position for list boxes with the sort property set to True. If the list box has its Sort property set to False, the string is added to the end of the list. Otherwise, the string is inserted into the list, and the list is sorted.

To insert a string into a specific location within the list, use the InsertString method.

```
int AddString( LPCTSTR lpszString );
```

lpszString	Points to the null-terminated string that is to be added.

If the return value is greater than or equal to 0, it is the zero-based index to the string in the list box. The return value is -1 if an error occurs; the return value is -2 if insufficient space is available to store the new string.

Cancel

The Cancel method can be used to end an Address Book session. The syntax of Cancel is

```
object.Cancel
```

where *object* is the name of a valid object.

CanPaste

The CanPaste method permits Paste operations. The syntax of CanPaste is as follows:

```
Boolean  CanPaste( )
```

CanUndo

The CanUndo method permits Undo operations. The syntax of CanUndo is as follows:

```
Boolean  CanUndo( )
```

Clear

Clear removes all objects from a containing control.

The syntax for the Clear method consists of:

```
object.Clear
```

object	The required term. Insert the name of a valid object.

ClearSel

The ClearSel method deletes (clears) the current selection, if any, in the edit control of the combo box.

The syntax for the Clear method consists of:

```
object.ClearSel
```

object	The required term. Insert the name of a valid object.

Copy

Copy copies the contents of a control to the Clipboard. (The contents are not deleted from the control.)

The syntax for the Copy method consists of:

```
object.Copy
```

object	The required item. Insert the name of a valid object.

The contents copied to the Clipboard depend on the control. For example, in a text field, the Copy method copies the currently selected text.

Cut

Cut deletes selected information from a control and copies it to the Clipboard.

The syntax for the Cut method consists of:

```
object.Cut
```

object	The required term. Insert the name of a valid object.

Delete

The Delete method deletes the contents of a control.

The syntax for the Delete method consists of:

```
object.Delete
```

object	The required term. Insert the name of a valid object.

DeleteString

The DeleteString method deletes a string from the list box of a combo box.

```
int DeleteString( INT nIndex );
```

nIndex	Specifies the index to the string that is to be deleted.

If the return value is greater than or equal to 0, then it is a count of the strings remaining in the list. The return value is -1 if nIndex specifies an index greater then the number of items in the list.

Dial

The Dial method attempts to launch the default TAPI dialer installed on the machine and dial the number specified in the edit section of the control.

The syntax for the Dial method consists of:

```
object.Dial
```

object	The required term. Insert the name of a valid object.

DialError event

The DialError event is fired when the specified number cannot be dialed.

```
Integer DialError( ERROR CODE )
```

Dir

The Dir method adds a list of filenames to the list box of a combo box.

```
int Dir( INT attr, String WildCard );
```

attr	Can be any combination following values:
&H0000	File can be read from or written to.
&H0001	File can be read from but not written to.
&H0002	File is hidden and does not appear in a directory listing.
&H0004	File is a system file.
&H0010	The name specified by lpszWildCard specifies a directory.
&H0020	File has been archived.
&H4000	Include all drives that match the name specified by lpszWildCard.
&H8000	Exclusive flag. If the exclusive flag is set, only files of the specified type are listed. Otherwise, files of the specified type are listed in addition to “normal” files.
WildCard	Points to a file-specification string. The string can contain wildcards (for example, .).

If the return value is greater than or equal to 0, it is the zero-based index of the last filename added to the list. The return value is -1 if an error occurs; the return value is -2 if insufficient space is available to store the new strings.

EmptyUndoBuffer

The EmptyUndoBuffer method empties the buffer set aside for storing data required to complete an Undo operation.

```
void    EmptyUndoBuffer( )
```

ExecError

The ExecError event is fired when the specified program, or the program registered for the specified filename extension, cannot be launched.

```
Integer    ExecError( ERROR CODE )
```

FindString

The FindString method finds (without selecting) the first string that contains the specified prefix in the list box of a combo box.

```
int FindString( int nStartAfter, String searchString )
const;
```

nStartAfter	Contains the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by nStartAfter. If –1, the entire list box is searched from the beginning.
searchString	Points to the null-terminated string that contains the prefix to search for. The search is case independent, so this string can contain any combination of uppercase and lowercase letters.

If the return value is greater than or equal to 0, it is the zero-based index of the matching item. It is -1 if the search was unsuccessful.

FindStringExact

The FindStringExact method finds the first list-box string (in a combo box) that matches a specified string.

```
int FindStringExact( int nIndexStart, String
searchString) const;
```

nIndexStart Specifies the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by nIndexStart. If nIndexStart is –1, the entire list box is searched from the beginning.

searchString Points to the null-terminated string to search for. This string can contain a complete filename, including the extension. The search is not case sensitive, so this string can contain any combination of uppercase and lowercase letters.

The return value is the zero-based index of the matching item, or -1 if the search was unsuccessful.

GetCount

The GetCount method retrieves the number of items in a list box or the list box part of a combo box.

```
int GetCount( ) const;
```

Returns the number of items. The returned count is one greater than the index value of the last item (the index is zero-based). It is -1 if an error occurs.

GetCurSel

The GetCurSel method retrieves the index of the currently selected item (if any) in a list box or the the list box part of a combo box.

```
int GetCurSel( ) const;
```

The return value is the zero-based index of the currently selected item in the list box of a combo box, or -1 if no item is selected.

GetDroppedState

The GetDroppedState method establishes whether a list box or the list box part of a drop-down combo box is visible (dropped down).

```
BOOL GetDroppedState( )
```

The return value is non-zero if the list box is visible; otherwise 0.

GetEditSel

The GetEditSel method gets the starting and ending character positions of the current selection in the edit control of a combo box.

```
Integer GetEditSel( ) const;
```

The return value is a 32-bit value that contains the starting position in the low-order word and the position of the first nonselected character after the end of the selection in the high-order word. If this function is used on a combo box without an edit control, -1 is returned.

GetItemData

The GetItemData method retrieves the application-supplied 32-bit value associated with the specified combo-box item.

```
integer GetItemData( int nIndex ) const;
```

nIndex	Contains the zero-based index of an item in the combo box's list box.

The return value is the 32-bit value associated with the item, or -1 if an error occurs.

GetItemText

The GetItemsTextmethod gets a string from a list box.

```
string GetText( int nIndex );
```

nIndex	Specifies the zero-based index of the string to be retrieved

The return value is the string at the position specified. Returns an empty string if the index is invalid.

GetItemTextLen

The GetItemTextLen method gets the length of a string in a list-box item.

```
int GetTextLen( int nIndex );
```

nIndex	Specifies the zero-based index of the string

The return value is the length of the string in bytes, excluding the terminating null character. If nIndex does not specify a valid index, the return value is -1.

GetLBText

The GetLBText method gets a string from the list box of a combo box.

```
string GetLBText( int nIndex )
```

nIndex	Contains the zero-based index of the list-box string to be copied.

The return value is the text contained in the list box index specified. If the index is invalid, it will return an empty string.

GetLBTextLen

The GetLBTextLen method gets the length of a string in the list box of a combo box.

```
int GetLBTextLen( int nIndex ) const;
```

nIndex	Contains the zero-based index of the list-box string.

The return value is the length of the string in bytes, excluding the terminating null character. If nIndex does not specify a valid index, the return value is -1.

GetLineFromChar

The GetLineFromChar method returns the number of the line containing a specified character position in the control

```
object.GetLineFromChar(charpos)
```

object	Required. An object expression that evaluates to a Rich Text control.
charpos	Required. A long integer that specifies the index of the character whose line you want to identify. The index of the first character in the Rich text control is 0.

You can use the GetLineFromChar method to find out which line in the text of a Rich text control contains a certain character position in the text. You might need to do this because the number of characters in each line of text can vary, making it very difficult to find out which line in the text contains a particular character, identified by its position in the text.

GetSel

The GetSel method can be used to get the starting and ending character positions of the current selection (if any) in an edit control, using either the return value or the parameters. The positions are zero based.

```
object.GetSel( integer REF nStartChar, integer REF
nEndChar )
```

nStartChar	Reference to an integer that will receive the position of the first character in the current selection.
nEndChar	Reference to an integer that will receive the position of the first nonselected character past the end of the current selection.

GetSelCount

The GetSelCount method retrieves the total number of selected items in a multiple-selection list box.

```
int GetSelCount( );
```

The return value is the count of selected items in a list box. If the list box is a single-selection list box, the return value is -1.

GetSelItems

The GetSelItems method fills a variant with an array of integers that specifies the item numbers of selected items in a multiple-selection list box.

```
int GetSelItems( variant REF indexArray ) const;
```

indexArray	a variant that will receive an array of integers of the currently selected items.

The return value is the actual number of items placed in the buffer. If the list box is a single-selection list box, the return value is -1.

GetTopIndex

The GetTopIndex method returns the index of the first visible item in the list-box portion of the combo box.

```
int GetTopIndex( ) const;
```

The return value is the zero-based index of the first visible item in the list-box portion of the combo box if successful, -1 otherwise.

InsertString

The InsertString method inserts a string into the list box of a combo box. Unlike the AddString method, InsertString does not cause a list with the Sort property set to True to be sorted.

```
int InsertString( int nIndex, string newString );
```

nIndex	Contains the zero-based index to the position in the list box that will receive the string. If this parameter is –1, the string is added to the end of the list.
NewString	Points to the null-terminated string that is to be inserted.

The return value is the zero-based index of the position at which the string was inserted. The return value is -1 if an error occurs. The return value is -2 if insufficient space is available to store the new string.

IsComplete

The IsComplete method is used to determine if all non-optional characters in a mask have been filled.

This method can be used to force the user to completely fill in the mask before going on, by trapping the exit event, checking the return value of this method, and if incomplete, setting focus back to the control.

```
Boolean object.IsComplete
```

Returns True if all non optional characters in the mask have been filled by the user. If any non optional character are still blank, it will return False.

Launch

The Launch method attempts to launch the application associated with the text in the edit portion of the control. Calling this method is equivalent to the user hitting the launch button.

The syntax for the Launch method consists of:

```
object.Launch
```

object	The required term. Insert the name of a valid object.

LimitText

The LimitText method limits the length of the text that the user can enter into the edit control of a combo box.

LimitText only limits the text the user can enter. It has no effect on any text already in the edit control when the message is sent, nor does it affect the length of the text copied to the edit control when a string in the list box is selected.

```
BOOL LimitText( int nMaxChars );
```

nMaxChars	Specifies the length (in bytes) of the text that the user can enter. If this parameter is 0, the text length is set to 65,535 bytes.

The return value is non-zero if successful. If called for a combo box with the ComboStyle drop list or for a combo box without an edit control, the return value is -1.

Paste

Paste copies the contents of the Clipboard to a control. (The contents are not deleted from the Clipboard.) Information pasted into a text field is treated as text.

The syntax for the Paste method consists of:

```
object.Paste
```

object	The required term. Insert the name of a valid object.

Record

The Record method saves field changes to the Maximizer Address Book folder. The syntax for Record is as follows:

```
object.Record
```

object	Insert the name of a valid object (*e.g.,* TheView).

Refresh

Refresh redraws a list in a combo box or list box.

The syntax for the Refresh method consists of:

```
object.Refresh
```

object	The required term. Insert the name of the control to be redrawn.

RemoveAllHighlightedDates

The RemoveAllHighlightedDates method removes all highlighted dates.

```
void    RemoveAllHighlightedDates( )
```

RunClick

RunClick triggers the EventClick event as if the user actually clicked on the specified object. The syntax for RunClick is as follows:

```
object.Runclick
```

object	Insert the name of a valid object.

RunInitialize

RunInitialize triggers the EventInitialize event for the specified object. The syntax for RunClick is as follows:

```
object.RunInitialize
```

object	Insert the name of a valid object.

RunTerminate

RunTerminate triggers the EventTerminate event for the specified object. The syntax for RunClick is as follows:

```
object.RunTerminate
```

object	Insert the name of a valid object.

SelectAll

The SelectAll method selects all text within the control.

```
void    SelectAll( )
```

SelectString

The SelectString method searches for a string in the list box of a combo box and, if the string is found, selects the string in the list box and copies the string to the edit control.

```
int SelectString( int nStartAfter, String selString );
```

nStartAfter	Contains the zero-based index of the item before the first item to be searched. When the search reaches the bottom of the list box, it continues from the top of the list box back to the item specified by nStartAfter. If –1, the entire list box is searched from the beginning.
selString	Points to the null-terminated string that contains the prefix to search for. The search is case independent, so this string can contain any combination of uppercase and lowercase letters.

The return value is the zero-based index of the selected item if the string was found. If the search was unsuccessful, the return value is -1 and the current selection is not changed.

SelItemRange

The SelItemRange method selects multiple consecutive items in a multiple-selection list box.

Use this method only with multiple-selection list boxes. If you need to select only one item in a multiple-selection list box, use SetSel instead.

```
int SelItemRange( BOOL bSelect, int nFirstItem, int
nLastItem );
```

bSelect	Specifies how to set the selection (if bSelect is TRUE, the string is selected and highlighted; if FALSE, the highlight is removed and the string is no longer selected)
nFirstItem	Specifies the zero-based index of the first item to set
nLastItem	Specifies the zero-based index of the last item to set

The return value is -1 if an error occurs.

SelPrint

If you use the Printer object as the destination of the text from the rich text control, you must first initialize the device context of the Printer object by printing something like a zero-length string.

The SelPrint method prints formatted text in the control.

```
object.SelPrint(hdc)
```

object	An object expression that evaluates to a Rich Text control
hdc	The device context of the device you plan to use to print the contents of the control

If text is selected in the rich text control, the SelPrint method sends only the selected text to the target device. If no text is selected, the entire contents of the rich text are sent to the target device.

The SelPrint method does not print text from the rich text control. Rather, it sends a copy of formatted text to a device that can print the text. For example, you can send the text to the Printer object using code as follows:

```
RichText1.SelPrint(Printer.hDC)
```

Notice that the hDC property of the Printer object is used to specify the device context argument of the SelPrint method.

SendMail

The SendMail method attempts to launch the default mail application and create a new email message addressed to the person specified in the edit section of the control. It is equivalent to the user hitting the send mail button on the control.

The syntax for the SendMail method consists of:

```
object.SendMail
```

object	The required term. Insert the name of a valid object.

SetCurSel

The SetCurSel method selects a string in a list box or the list box part of a combo box. If necessary, the list box scrolls the string into view (if the list box is visible). The text in the edit control of the combo box is changed to reflect the new selection. Any previous selection in the list box is removed.

```
integer SetCurSel( int nSelect );
```

nSelect	Specifies the zero-based index of the string to select. If –1, any current selection in the list box is removed and the edit control is cleared.

The return value is the zero-based index of the item selected if the message was successful. The return value is -1 if nSelect is greater than the number of items in the list or if nSelect is set to –1, which clears the selection.

SetEditableSel

The SetEditableSel method attempts to set the cursor at the first editable character found before or after the position specified.

The return value is True if an editable character was found and selected, False if it could not find an editable chracter on or after/ before the position specified.

```
Boolean SetEditableSel( int nPos, Boolean bAfter );
```

nPos	Specifies the zero-based position of the character to select.
Bafter	If True, the cursor will be set to the first editable character on or after the position specified. If False, the cursor will be set to the first editable character on or before the position specified (working from right to left).

SetEditSel

The SetEditSel method selects characters in the edit control of a combo box.

```
BOOL SetEditSel( int nStartChar, int nEndChar );
```

nStartChar	Specifies the starting position. If the starting position is set to –1, then any existing selection is removed.
nEndChar	Specifies the ending position. If the ending position is set to –1, then all text from the starting position to the last character in the edit control is selected.

The return value is non-zero if the method is successful; otherwise 0. It is -1 if the combo box has the drop List ComboStyle or does not have a list box.

SetItemData

The SetItemData method sets the 32-bit value associated with the specified item in a combo box.

```
int SetItemData( int nIndex, DWORD dwItemData );
```

nIndex	Contains a zero-based index to the item to set.
dwItemData	Contains the new value to associate with the item.

The return value is -1 if an error occurs.

SetNow

The SetNow method sets the clock to the current time.

```
voidSetNow( )
```

SetSel

The SetSel method can be used to select a range of characters in an edit control.

```
SetSel( int nStartChar, int nEndChar)
```

nStartChar	Specifies the starting position. If nStartChar is 0 and nEndChar is –1, all the text in the control is selected. If nStartChar is –1, any current selection is removed.
nEndChar	Specifies the ending position.

SetToday

The SetToday method sets the calendar to today's date.

```
voidSetToday( )
```

SetTopIndex

The SetTopIndex method tells the list-box portion of the combo box to display the item with the specified index at the top.

The system scrolls the list box until either the item specified by nIndex appears at the top of the list box or the maximum scroll range has been reached.

```
int SetTopIndex( int nIndex );
```

nIndex	Specifies the zero-based index of the list-box item.

The return value is zero if successful, or -1 if an error occurs.

ShowDropDown

The ShowDropDown method shows or hides the list box of a combo box that has the ComboStyle of drop down or drop list.

```
ShowDropDown( BOOL bShowIt = TRUE );
```

bShowIt	Specifies whether the drop-down list box is to be shown or hidden. A value of TRUE shows the list box. A value of FALSE hides the list box.

SizeToFit

The SizeToFit method resizes the control so that everything fits properly given the current font properties.

```
voidSizeToFit( )
```

Span

The Span method selects text based on a set of specified characters.

```
object.Span characterset, forward, negate
```

object	Required. An object expression that evaluates to a Rich Text control.
characterset	Required. A string expression that specifies the set of characters to look for when extending the selection, based on the value of negate.
forward	Optional. A Boolean expression that determines which direction the insertion point moves, as described in Settings.
negate	Optional. A Boolean expression that determines whether the characters in the character set define the set of target characters or are excluded from the set of target characters, as described below.

The settings for forward are:

True	(Default) Selects text from the current insertion point or the beginning of the current selection forward, toward the end of the text
False	Selects text from the current insertion point or the beginning of the current selection backward, toward the start of the text

The settings for negate are:

True	The characters included in the selection are those that do not appear in the characterset argument. The selection stops at the first character found that appears in the characterset argument.
False	(Default) The characters included in the selection are those that appear in the characterset argument. The selection stops at the first character found that does not appear in the characterset argument.

The Span method is primarily used to easily select a word or sentence in the rich text control.

If the Span method cannot find the specified characters based on the values of the arguments, then the current insertion point or selection remains unchanged.

The Span method does not return any data.

Start

The Start method stops the clock. The clock will update itself every minute.

```
voidStart( )
```

Stop

The Stop method stops the clock.

```
voidStop( )
```

Undo

Undo undoes the last edit operation

The syntax for the Undo method consists of:

```
object.Undo
```

object	The required term. Insert the name of the control to be redrawn.

UpTo

The UpTo method moves the insertion point up to (but not including) the first character that is a member of a specified character set.

```
object.Upto(characterset, forward, negate)
```

object	Required. An object expression that evaluates to a Rich Text control.
characterset	Required. A string expression that specifies the set of characters to look for when moving the insertion point, based on the value of negate.

forward	Optional. A Boolean expression that determines which direction the insertion point moves, as described in Settings.
negate	Optional. A Boolean expression that determines whether the characters in the character set define the set of target characters or are excluded from the set of target characters, as described below.

The settings for forward are:

True	(Default) Moves the insertion point forward, toward the end of the text.
False	Moves the insertion point backward, toward the start of the text.

The settings for negate are:

True	The characters not specified in the characterset argument are used to move the insertion point.
False	(Default) The characters specified in the characterset argument are used to move the insertion point.

Common Events

Form Designer objects can be configured to respond to events. Certain events are common to all objects. The following events are applicable to the Maximizer Form Designer objects:

Common Events	Description
EventPumpData	C++ or VBScript action callback procedure executed when new data is pumped to the frame
EventClick	C++ or VBScript action callback procedure executed when a user clicks over the object (or for a Windows control, when it changes value)

EventChange	C++ or VBScript action callback procedure executed when a user clicks over the object (or for a Windows control, when it changes value)
EventInitialize	This event is fired when the TheFrame/ TheView or an object is loaded. Any code contained within this event is executed before any data processing commences.

CHAPTER 5

Properties

Change the Appearance of Form Designer Objects

In this chapter...

Common Properties

The following properties are applicable to all Maximizer Form Designer objects.

There are various other properties unique to individual controls—these and the stock properties are in addition to properties which may be available in the general programming environment.

(ObjectCode)

(ObjectCode) identifies the object for convenient access.

This is used for referencing the object within the Enable Script. This way you can control and execute properties and methods of an object that is contained within TheView (*e.g.*, [ObjectName].[Property or Method])

Values

Object name and index. (*e.g.,* EditBox1)

Accelerator

The key accelerator (or *mnemonic*) that is selected allows quick access to the Object in Run Mode. To access the Object the user should use the combination of the ALT key and the accelerator.

Values

Empty (Default)

Alignment

Determines the position of the text contained within the label.

Values

0	Left
1	Center (Default)
2	Right

BackColor

Changes the color of the object's background to the selected windows color.

If set to Nil the the object's BackColor will be transparent.

Values

Nil (Default)

Bottom

The Bottom property defines the height of the object in pixels, and is more commonly known as the height of the object. Use the Bottom property for operations based on an object's Internal dimensions, such as resizing.

Values

Placing co-ordinate

Color

Changes the color of the form or object to the selected color value. If set to Nil, the form will use the system default color.

CursorPointer

Cursor pointer displayed while mouse is over object.

You can use this property when you want to indicate changes in functionality as the mouse pointer passes over an Object or Frame/View.

Values

0	(Default) Shape determined by the object.
1	Arrow.
2	I-beam. Used mainly for documents or text.
3	Cross (crosshair pointer).
4	Up Arrow.
5	No Drop. Used for drag/drop operation.
6	Help. Displays the cursor help symbol.
99	HotSpot

Enable

Enables the object's hotspots, event handling, and its state.

The Enabled property allows Objects to be enabled or disabled at Run Time. For example, you can disable Objects that don't apply to the current state of the TheFrame/TheView. You can also disable a control used purely for display purposes, such as an EditBox Object that provides read-only information.

Values

0	No
1	Yes

Font

For all Objects, the default font is Nil. When this property is selected, a font selection dialog will appear the currently selected font of the Object. By changing the values in this dialog will update the Object accordingly.

Values

Nil (Default)

Group

Controls whether or not the control is the first in a group of controls, such as a group of radio buttons.

Values

0	No
1	Yes

HelpContextID

Used if the frame object's WhatsThisHelp property is set to 1 - Yes.

Values

0 (Default)

HelpHotButton

Displays a "?" Help button next to the control.

Values

0	Yes
1	No

Left

The Left property defines position of the object as measured in pixels from the left edge of the frame. Use the Left property for operations based on an object's external dimensions, such as moving.

Values

Placing co-ordinate

MaximumEq

The value set here determines the highest value that can be entered into this Object.

Values

100 (Default)

MinimumEq

The value set here determines the lowest value that can be entered into this Object.

Values

0 (Default)

Right

The Right property defines the width of the object in pixels. Use the Right property for operations based on an object's Internal dimensions, such as resizing.

Values

Placing co-ordinate

Tabstop

Indicates whether a user can use the Tab key to give the focus to the object.

This property enables you to add or remove an Object from the tab order on TheView. For example, if you're using a Bitmap Object to display a bitmap, set its TabStop property to False, so the user can't tab to the Bitmap Object.

Values

0	No
1	Yes

TextColor

Changes the color of the Object's text to the selected windows color.

If set to Nil the Object's text color will default to the Windows text color.

Values

Nil (Default)

ToolTipText

Text displayed while the mouse is over object.

This can be helpful in providing a better description of the object during Run Time.

Values

Empty (Default)

Top

The Top property defines the position of the objects as measured in pixels from the top edge of the frame. Use the Top property for operations based on an object's external dimensions, such as moving.

Values

Placing co-ordinate

ValueID

Within this property you can select from a defined list of local variables declared in TheFrame/TheView property LocalVars, which is then attached to the Object. Any updating of the local variable selected will be reflected in the Object(s) the it is attached with. The value is formatted and displayed at Run Time.

Values

(None) (Default)

Visible

Determines whether an object or control is visible.

To hide an Object at displaying of a form, set the Visible property to 0 - No at Design Time. Setting this property in Enable Script enables you to hide and later redisplay an Object at Run Time in response to a particular event.

Values

0	No
1	Yes

Other Properties

AdjacentMonthTextColor

AdjacentMonthTextColor specifies the text color used for the months preceding and following the current month.

The syntax of the AdjacentMonthTextColor property consists of:

```
object.AdjacentMonthTextColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the color of the text.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue.

AlignTextLeft

AlignTextLeft is a property for CheckButton and RadioButton controls that defines whether the label text appears to the left or right of the button. The syntax for the AlignTextLeft property consists of:

```
object.AlignTextLeft [ = Boolean ]
```

object	Insert the name of a valid object.
Boolean	A boolean value specifying whether or not the text is aligned left or right.

Possible values for Boolean are as follows:

No	Text appears to the right of the button.
Yes	Text appears to the left of the button.

The default value for AlignTextLeft is No.

AllowNoSetTime

AllowNoSetTime defines whether the user can change the time value in the DateTimePicker control. Possible values for this control are as follows:

0	No. User may not set the time value in the DateTimePicker.
1	Yes. User may set the time value in the DateTimePicker.

The default value for AllowNowSetTime is No.

AMPMChange

AMPMChange specifies the number of minutes from midnight when AM switches to PM.

The syntax for the AMPMChange property consists of:

```
object.AMPMChange [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value specifying the number of minutes after midnight.

The default value for AMPMChange is 720.

AnchorSnaps

The AnchorSnaps property defines the locations of Snap Points within the object. Objects that have the AnchorSnaps property are: Text, TextVar, and Bitmap. To define Snap Points, click on the AnchorSnaps property to open the Enter Snap Points dialog. For more information on using Snap Points, please see the Chapter entitled, "Working with Forms".

ArrowHead

Defines the style of the Line object, using arrow heads. ArrowHead has four possible values.

Values

0	No arrow
1	←Left arrow
2	→Right arrow
3	↔ Both arrows

ArrowHeadHeight

Defines the size of the arrow head for the Line object. This property is only applicable when the ArrowHead property has an arrowhead value selected.

AutoHideToolBar

AutoHideToolBar specifies whether the toolbar associated with the Rich Text control is displayed only when the control has the focus.

AutoHideToolBar is ignored if the ToolBar property is set to False.

The syntax of the AutoHideToolBar property consists of:

```
object.AutoHideToolBar [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the toolbar appears only when the control has the focus.

The possible values for Boolean are:

True	The toolbar appears only when its control has the focus.
False	The toolbar is always visible.

The default value for the AutoHideToolBar property is True (the toolbar is visible only when its control has the focus).

AutoRecord

Defines whether changes in Maximizer Form Designer are written directly to Maximizer as they occur or after they occur. The possible values for AutoRecord are as follows:

0	No. Does not write changes directly to Maximizer. You will be prompted to save any changes when the form is closed if no Record method was called.
1	Yes. Writes changes directly to Maximizer as the changes occur.
2	Detach. Will not prompt you to save your changes, and does not automatically write changes to Maximizer.

BackgroundColor

BackColor specifies the background color of a control. The syntax of the BackColor property consists of:

```
object.BackColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue.

Bitmap

The Bitmap property defines the path and file name of a bitmap, which can be displayed on the face of a Button object. When a a file name is specified in the Bitmap property, the bitmap is displayed "behind" the button's text label.

The syntax of the Bitmap property is as follows:

```
object.Bitmap [ = String ]
```

object	Insert the name of a valid object.
String	A string containing the path and file name of the bitmap file.

The default value of the Bitmap property is (None).

Blank

Blank specifies whether a control contains no valid date or time. The syntax of the Blank property consists of:

```
object.Blank [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the control contains no valid date or time.

The possible values for Boolean are:

True	The control contains no valid date or time.
False	The control contains a valid date or time.

The default value for the Blank property is False (the control contains a valid date or time).

BlankCharacter

BlankCharacter specifies a character used to signify a character position that has no data.

The syntax of the BlankCharacter property consists of:

```
object.BlankCharacter [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that specifies the character used to signify a character position that has no data.

The default setting for BlankCharacter is '_'.

BorderColor

BorderColor defines the color of the control. For BorderColor or BorderWidth to take effect, the BorderStyle property must have a value of 1 or 2.

The syntax of the BorderColor property consists of:

```
object.BorderColor [color]
```

object	Insert the name of a valid object.
color	Insert a valid color code.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

BorderDrawn

The BorderDrawn property defines whether or not the control has a border. The syntax of BorderDrawn consists of:

```
object.BorderDrawn [ = Boolean ]
```

object	Insert the name of a valid object.
Boolean	Insert a boolean value controlling whether or not the border is drawn on the control.

The default value of BorderDrawn is 1 – Yes.

BorderStyle

BorderStyle specifies the type of border (none or single-line) used by a control. The syntax of the BorderStyle property consists of:

```
object.BorderStyle [ = trkBorderStyle ]
```

object	The required term. Insert the name of a valid object.
trkBorderStyle	The optional term. An integer constant that specifies the border style.

Possible values for trkBorderStyle are:

0	trkBorderStyleNone—the control has no visible border.
1	trkBorderStyleSingle—the control has a single-line border.

The default value for BorderStyle is 0 (no border).

BorderWidth

Defines the border width in pixels for the controls that support the BorderWidth property.

BulletIndent

BulletIndent specifies the amount by which text is indented in a Rich Text control.

The syntax of the BulletIndent property consists of:

```
object.BulletIndent [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer that determines the amount of indent in the Rich Text control.

ButtonHeight

ButtonHeight specifies the height (in pixels) of the Month and Year buttons.

The syntax of the ButtonHeight property consists of:

```
object.ButtonHeight [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the height (in pixels) of Month and Year buttons.

ButtonShape

Defines the style of the Button control. Buttons can have one of three ButtonShape values:

0	Normal
1	Property Tab (Inactive)—sunken (not selected).
2	Property Tab (Active)—raised (selected).

When used in conjunction with the Layer property, you can use Property Tab values to create tab-style form without using a Tab Control object. See the section on using layers in Chapter 2 for more information.

ButtonType

Defines the action taken when the button is clicked. Buttons can be one of 5 values:

0	Cancel. A standard Cancel button, which ends the form.
1	EventClick. Calls the EventClick method.
2	Goto.
3	Help. Brings up help using the context ID defined in the HelpContextID property.
4	OK. A standard OK button, which validates the field data in the form.
5	Record. Saves changes to Maximizer.

ButtonWidth

ButtonWidth specifies the width (in pixels) of the Month and Year buttons.

The syntax of the ButtonWidth property consists of:

```
object.ButtonWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) of Month and Year buttons.

CalendarBackColor

CalendarBackColor defines the background color of the calendar in the DateTimePicker control.

The syntax of the CalendarBackColor property consists of:

```
object.CalendarBackColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

CalendarButtonWidth

CalendarButtonWidth specifies the width (in pixels) of the Calendar button in the DateTimePicker control.

The syntax of the CalendarButtonWidth property consists of:

```
object.CalendarButtonWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) of the Calendar button.

CalendarHeight

CalendarHeight specifies the height (in pixels) of the Calendar popup in the DateTimePicker control.

The syntax of the CalendarHeight property consists of:

```
object.CalendarHeight [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the height (in pixels) of the Calendar popup.

CalendarWidth

CalendarWidth specifies the width (in pixels) of the Calendar popup in the DateTimePicker control.

The syntax of the CalendarWidth property consists of:

```
object.CalendarWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) of the Calendar popup.

Caption

Caption provides descriptive text that appears on a control to identify or explain it.

The ForeColor property of the control determines the color of the text in the caption. If a control's caption is too long, the caption is truncated.

The syntax of the Caption property consists of:

```
object.Caption [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that specifies the text that will be displayed as the caption.

The default setting for the caption of a new control on a form is the name of the control.

CaseOrPassword

Defines the format in which the information is displayed withing the edit control. Values for CaseOrPassword can be one of the following:

0	None. No formatting is used.
1	Lower case. All characters are forced to lower case.
2	Upper case. All characters are forced to upper case.
3	Password. Characters display as asterisks "*" instead of the typed character.

Check

Check specifies the "check state" (selected state) of a radio button.

The syntax for the Check property consists of:

```
object.Check [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the radio button is checked or unchecked.

The possible values for Boolean are:

True	The button is checked.
False	The button is unchecked.

The default value for Check is False (the button is unchecked).

CheckedValue

CheckedValue specifies the value used for data awareness when the control is checked. The properties CheckedValue and UncheckedValue together can be used as a Boolean pair. The property defaults to 'True' but can be set to other values such as 'Paid' and 'Unpaid' etc.

The strings set in the CheckedValue and UncheckedValue properties are returned as a variant in the value property, depending upon the checked state. Since Value is a variant, it can be used as any type that the variant conversion functions support, Hence the with the default strings as 'True' and 'False', the Value property can be used as a Boolean value.

The syntax of the CheckedValue property consists of:

```
object.CheckedValue [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that is used for data awareness when the check box is "checked".

The default value for CheckedValue is True (the check box is checked).

ClockButtonWidth

ClockButtonWidth specifies the width (in pixels) of the Clock button in the DateTimePicker control.

The syntax of the ClockButtonWidth property consists of:

```
object.ClockButtonWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) of the Clock button.

ClockHeight

ClockHeight specifies the height of the Clock popup in the DateTimePicker control.

The syntax of the ClockHeight property consists of:

```
object.ClockHeight [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the height (in pixels) of Clock popup.

ClockOffset

ClockOffset specifies the offset (in minutes) to be added to the current time when the clock is running. (See the Start and Stop methods.) This enables you to set a clock that displays a non-local time (perhaps next to one that does).

The syntax for the ClockOffset property consists of:

```
object.ClockOffset [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value specifying the offset to be added to the current time.

The default value for ClockOffset is zero (0).

ClockWidth

ClockWidth specifies the width of the Clock popup in the DateTimePicker control.

The syntax of the ClockWidth property consists of:

```
object.ClockWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) of Clock popup.

ColorFriday

ColorFriday specifies the text color for dates falling on a Friday.

The syntax of the ColorFriday property consists of:

```
object.ColorFriday [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ColorMonday

ColorMonday specifies the text color for dates falling on a Monday.

The syntax of the ColorMonday property consists of:

```
object.ColorMonday [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ColorSaturday

ColorSaturday specifies the text color for dates falling on a Saturday.

The syntax of the ColorSaturday property consists of:

```
object.ColorSaturday [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ColorSunday

ColorSunday specifies the text color for dates falling on a Sunday.

The syntax of the ColorSunday property consists of:

```
object.ColorSunday [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ColorThursday

ColorThursday specifies the text color for dates falling on a Thursday.

The syntax of the ColorThursday property consists of:

```
object.ColorThursday [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function,

which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ColorTuesday

ColorTuesday specifies the text color for dates falling on a Tuesday.

The syntax of the ColorTuesday property consists of:

```
object.ColorTuesday [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ColorWednesday

ColorWednesday specifies the text color for dates falling on a Wednesday.

The syntax of the ColorWednesday property consists of:

```
object.ColorWednesday [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ColumnWidth

ColumnWidth specifies the width (in pixels) of all columns in a multi-column list box.

The syntax of the ColumnWidth property consists of:

```
object.ColumnWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) of all columns in a multi-column list box.

The default value for ColumnWidth is 50.

ComboStyle

ComboStyle specifies whether a combo box appears as a simple combo box, a drop-down combo box (where you can type a value in a text box or selecting one from a drop-down list), or a drop-list combo box (where you must select a value from the list).

The syntax of the ComboStyle property consists of:

```
object.ComboStyle [ = trkComboStyle ]
```

object	The required term. Insert the name of a valid object.
trkComboStyle	The optional term. An integer constant specifying the kind of combo box.

Possible values for trkComboStyle are:

0	The control is a simple combo box
1	The control is a drop-down combo box
2	The control is a drop-list combo box

The default value for ComboStyle is 1 (a drop-down combo box).

ComboType

ComboType defines the functionality of a combo box control. The values of ComboType can be one of the following:

0	Dropdown
1	Droplist

Container

Container specifies the container of a control on a Form. (A container is a control that can contain other controls.)

The syntax of the Container property consists of:

```
object.Container [ = Container ]
```

object	The required term.. Insert the name of a valid object.
Container	The optional term. A type of control that can serve as a container for other controls.

ContextMenu

ContextMenu enables the context menu associated with a control.

The syntax of the ContextMenu property consists of:

```
object.ContextMenu [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the context menu is enabled.

The possible values for Boolean are:

True	The context menu is enabled.
False	The context menu is disabled.

The default value for ContextMenu is True (the context menu is enabled).

DataChanged

DataChanged indicates if the data in a control has been changed from the data retrieved from the current record.

The syntax of the DataChanged property consists of:

```
object.DataChanged [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether data in the control has changed.

Possible values for Boolean are:

True	The data has changed.
False	The data has not changed.

The default value for DataChanged is False (the data has not changed).

DataSourceValue

DataSourceValue specifies the value associated with a radio button for data awareness. If the radio button is checked, this value will be written to the Address Book folder. This value should be unique within the radio group.

The syntax for the DataSourceValue property consists of:

```
object.DataSourceValue [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that specifies the value associated with the radio button for data awareness.

The default value for DataSourceValue is the name of the selected OLE control.

DatesFont

DatesFont specifies the font used for dates.

The syntax of the DatesFont property consists of:

```
object.DatesFont [ = Font ]
```

object	The required term. Insert the name of a valid object.
Font	The optional term. The font object used for dates.

Day

Day specifies the day of month that is currently selected.

The syntax of the Day property consists of:

```
object.Day [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The currently selected day of the month.

Day can take a value in the range 1 to 31.

DecimalBase

The DecimalBase property of the Spinner object determines whether the spinner value is in decimal or hexidecimal. The syntax of DecimalBase consists of:

```
object.DecimalBase [ = Boolean ]
```

object	Insert the name of a valid Spinner object.
Boolean	Determines whether the spinner value is decimal or hexidecimal.

The default value is 1 – Yes (Decimal).

DecimalPlaces

DecimalPlaces specifies the number of decimal places that may be typed into a control.

The syntax of the DecimalPlaces property consists of:

```
object.DecimalPlaces [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The number of decimal places that may be typed into the control.

If DecimalPlaces is negative, no limit is placed on the number of decimal places. If it is a value greater than or equal to 0, the control will allow only this number of decimal places to be typed in.

This does not stop more decimal places being set to the control through a programming call such as setting the Value property.

The default value for DecimalPlaces is -1 (there is no limit on the number of decimal places).

DisableNoScroll

DisableNoScroll specifies whether scroll bars in the Rich Text control are disabled. DisableNoScroll is ignored when the ScrollBars property is set to zero (0).

The syntax of the DisableNoScroll property consists of:

```
object.DisableNoScroll [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. It specifies whether scroll bars are enabled in a Rich Text control.

The possible values for Boolean are:

True	Scroll bars are disabled.
False	Scroll bars are enabled.

The default value for DisableNoScroll is False (the scroll bars are enabled).

DragBehavior

DragBehavior specifies whether a control supports “drag-and-drop” behavior.

The syntax of the DragBehavior property consists of:

```
object.DragBehavior [ = trkDragBehavior ]
```

object	The required term. Insert the name of a valid object.
trkDragBehavior	The optional term. A constant that specifies whether the drag-and-drop feature is enabled.

Possible values for trkDragBehavior are:

0	trkDragBehaviorDisabled. A drag-and-drop action is not allowed.
1	trkDragBehaviorEnabled. A drag-and-drop action is allowed.

If the DragBehavior property is enabled, dragging in a text field starts a drag-and-drop operation on the selected text.

If DragBehavior is disabled, dragging in a text field or combo box selects text. The drop-down part of a combo box does not support "drag-and-drop", nor does it support selection of list items within the text.

The default value for DragBehavior is 0 (drag-and-drop is not allowed).

DropDownRows

DropDownRows specifies the number of rows in the list box section of a combo box.

The syntax of the DropDownRows property consists of:

```
object.DropDownRows [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the number of rows in the list box section of a drop-down or drop-list combo box.

The number of rows must be greater than or equal to zero (0). The default value is 2.

DroppedWidth

DroppedWidth specifies the minimum allowable width (in pixels) for the drop-down list-box section of a combo box.

The syntax of the DroppedWidth property consists of:

```
object.DroppedWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) of the list box section of a drop-down or drop-list combo box.

The default value for DroppedWidth is -1.

Enabled

Enabled specifies whether a control is enabled (that is, it can receive the focus and respond to mouse or keyboard events) or disabled. (Visually, a disabled control appears dimmed; an enabled control does not.)

The syntax of the Enabled property consists of:

```
object.Enabled [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the control can respond to mouse or keyboard events.

The possible values for Boolean are:

True	The control is enabled. It can receive the focus and respond to mouse or keyboard events. The control is accessible through code.
False	The control is disabled, and you cannot interact with it. The control is usually still accessible through code.

In combination, the Enabled and Locked properties do the following:

- If Enabled and Locked are both True, the control can receive the focus, and you can copy, but not edit, data in the control.
- If Enabled is True and Locked is False, the control can receive the focus, and you can both copy and edit data in the control.
- If Enabled is False, the value of Locked is ignored. The control cannot receive the focus, and you can neither copy nor edit data in the control.

The default value of Enabled is True (the control can receive the focus and respond to mouse or keyboard events).

EnableMaximum

EnableMaximum specifies whether the Maximum property is used to limit the maximum value that can be typed into the control.

The syntax of the EnableMaximum property consists of:

```
object.EnableMaximum [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the Maximum property is used to limit the maximum value.

The possible values for Boolean are:

True	The Maximum property is used to limit the maximum value in the control.
False	The Maximum property is ignored.

The default value for EnableMaximum is False (the Maximum property is ignored).

EnableMinimum

EnableMinimum specifies whether the Minimum property is used to limit the minimum value that can be typed into the control.

The syntax of the EnableMinimum property consists of:

```
object.EnableMinimum [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the Minimum property is used to limit the minimum value.

The possible values for Boolean are:

True	The Minimum property is used to limit the minimum value in the control.
False	The Minimum property is ignored.

The default value for EnableMinimum is False (the Minimum property is ignored).

ExtendedUI

ExtendedUI specifies whether a drop-down or drop-list combo box should use the extended user interface or the standard user interface.

The syntax of the ExtendedUI property consists of:

```
object.ExtendedUI [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether to use the extended user interface or the standard user interface.

The possible values for Boolean are:

True	Use the extended user interface.
False	Use the standard user interface.

When you use the extended user interface,

- pressing the Down Arrow key will show the control's list box;

- clicking the static control will show the list box only for drop-list combo boxes; and
- scrolling in the static control is disabled when the list is hidden.

The default value for ExtendedUI is True (use the extended user interface).

FileName

FileName specifies the name of a file that contains text in Rich Text format.

The syntax for the FileName property consists of:

```
object.FileName [ = String ]
```

object	The required term. Insert the name of a valid Rich Text control.
String	The optional term. Insert the string expression that specifies the path and name of the file containing the text that is in Rich Text format.

FirstDayOfWeek

FirstDayOfWeek specifies the day of the week that appears in the first column of the control. This affects week-numbering.

The syntax of the FirstDayOfWeek property consists of:

```
object.FirstDayOfWeek [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The integer value specifying the first day of the week.

The possible values for Integer are:

0	Sunday
1	Monday Default Value The default value for FirstDayOfWeek is 0 (Sunday).

FirstMonth

FirstMonth specifies the first month of the year for week number calculations.

The syntax of the FirstMonth property consists of:

```
object.FirstMonth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The integer value specifying the first month of the year.

FirstMonth can take a value in the range 1 to 12.

ForeColor

ForeColor specifies the foreground color of a control.

For a scroll bar or spin button, ForeColor sets the color of the arrows. For a label, ForeColor determines the color of the text.

```
object.ForeColor [ = Color ]
```

The syntax of the ForeColor property consists of:

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the foreground color of a control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

GotoPath

The GotoPath property defines the path and file to retrieve when the Button control's ButtonType property is set to Goto. The default value for GotoPath is None.

GroupName

GroupName specifies the name of a radio button group. Any radio buttons on the same form with the same GroupName become part of this group.

Only one radio button in a group can be "checked" (selected) at any one time. Selecting a radio button in a group will deselect all the others. The value of the group can be accessed via the RadioGroupValue property.

The syntax for the GroupName property consists of:

```
object.GroupName [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that specifies the name of the radio button group.

The default value for GroupName is an empty string (a string of zero length).

HandHighColor

HandHighColor specifies the light color of the Hour and Minute hands in the Clock control.

The syntax of the HandHighColor property consists of:

```
object.HandHighColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the color.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

HandLowColor

HandLowColor specifies the shadow color of the Hour and Minute hands in the Clock control.

The syntax of the HandLowColor property consists of:

```
object.HandLowColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the shadow color.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

HatchStyle

Defines the background style for the TextVar object. The possible HatchStyle values are as follows:

0	None.
1	Horizontal (----)
2	Vertical (|||||)
3	Diagonal (\\\\)
4	DiagRev (////)
5	Cross (++++)
6	DiagCross (xxxx)

The HatchStyle will display in the same color specified as the text ForeColor property.

HeadingFont

HeadingFont specifies the font used for the heading (the Month and Year text that appears between the buttons).

The syntax of the HeadingFont property consists of:

```
object.HeadingFont [ = Font ]
```

object	The required term. Insert the name of a valid object.
Font	The optional term. The font object used for the heading.

Height

The Height property defines the height of TheFrame in pixels. The syntax of the height property consists of:

```
TheFrame.Height [ = Long ]
```

Long	The height of TheFrame in pixels.

HideSelection

HideSelection specifies whether selected text remains highlighted when a control no longer has the focus.

The syntax of the HideSelection property consists of:

```
object.HideSelection [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the selected text remains highlighted when the control does not have the focus.

Possible values for Boolean are:

True	Selected text is not highlighted unless the control has the focus.
False	Selected text always appears highlighted.

The default value for HideSelection is True (selected text is only highlighted if it has the focus).

HighlightBackColor

HighlightBackColor specifies the background color for highlighted dates in both the Calendar and DateTimePicker controls.

The syntax of the HighlightBackColor property consists of:

```
object.HighlightBackColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

The default value for HighlightBackColor is gray.

HighlightColor

HighlightColor specifies the foreground text color for highlighted dates.

The syntax of the HighlightColor property consists of:

```
object.HighlightColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the foreground text color.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

HighlightForeColor

HighlightForeColor specifies the foreground text color for highlighted dates in the DateTimePicker control.

The syntax of the HighlightForeColor property consists of:

```
object.HighlightForeColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the foreground text color.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

The default value for HighlightForeColor is black.

HorizontalExtent

HorizontalExtent specifies the width (in pixels) by which the list-box section of the combo box can be scrolled horizontally.

The syntax of the HorizontalExtent property consists of:

```
object.HorizontalExtent [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the width (in pixels) by which the list box section of the combo box can be scrolled horizontally.

The horizontal scroll bar associated with this control will appear only if the width of the list box is smaller than the value specified in HorizontalExtent.

The default value for HorizontalExtent is zero (0). (A horizontal scroll bar will never appear.)

Hour

Hour specifies the hour of the day.

The syntax of the Hour property consists of:

```
object.Hour [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The hour of the day.

Hour can take a value in the range 0 to 23.

HourColor

HourColor specifies the color of clock text (digits or Roman numerals) on the Clock control.

The syntax of the HourColor property consists of:

```
object.HourColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the color of clock text.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

HScroll

HScroll enables or disables the horizontal scroll bar.

The syntax of the HScroll property consists of:

```
object.HScroll [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the combo box may have a horizontal scroll bar.

Possible values of Boolean are:

True	The horizontal scroll bar is enabled.
False	The horizontal scroll bar is disabled.

The default value for HScroll is False (the scroll bar is disabled).

hWnd

hWnd returns a handle to a control.

The syntax of the hWnd property consists of:

```
object.hWnd
```

object	The required term. Insert the name of a valid Rich Text control.

IncrAccelerator

IncrAccelerator defines the amount that a value stored in the control will be incremented or decremented when the control's up or down buttons are clicked.

Increment

Increment defines the amount that a value stored in the control will be incremented or decremented when the control's up or down buttons are clicked.

The syntax of the Increment property consists of:

```
object.Increment [ = Double ]
```

*object*The required term. Insert the name of a valid object.

*Double*The optional term. The amount by which the value in the control is incremented or decremented when spin buttons are clicked.

The default value for Increment is one (1).

KeyboardActive

KeyBoardActive specifies whether keyboard input is enabled or disabled.

The syntax of the KeyBoardActive property consists of:

```
object.KeyBoardActive [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the control can take input from the keyboard.

The possible values for Boolean are:

True	The keyboard is enabled.
False	The keyboard is disabled.

The default value for KeyBoardActive is True (use the extended user interface).

LargeButtonBitmap

LargeButtonBitmap specifies whether a 32x32 pixel bitmap is displayed on the Dial/Launch/Sendmail button associated with a control, or weather a 16x16 pixel bitmap will be used.

```
object.LargeButtonBitmap [ = Boolean ]
```

The syntax of the LargeButtonBitmap property consists of:

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the image displayed on the button is large or small.

The possible values for Boolean are:

True	A 32x32 pixel bitmap is displayed on the button.
False	A 16x16 pixel bitmap is displayed on the button.

The default value for LargeButtonBitmap is False (display a small bitmap).

Layer

Layer defines the layer to which the control is assigned. Layer can be used in conjunction with the Visible property to make layers of controls visible or not on certain events. Layers are defined in the Layers Sheet, which is accessed by selecting Layout > Layers from the Form Designer menu.

The default value for Layer is “Default”.

Layout

The Layout property defines how the bitmap image appears in the Bitmap object. The syntax for Layout consists of:

```
object.Layout [ = Integer ]
```

object	Insert the name of a valid Bitmap object.
Integer	An integer constant that specifies the kind of layout used by the bitmap.

Possible values for *Integer* are:

0	Center—the bitmap is centered in the Bitmap object.
1	Left—the left edge of the bitmap is aligned with the left edge of the Bitmap object.
2	Right—the right edge of the bitmap is aligned with the right edge of the Bitmap object.
3	Stretch—the bitmap is resized (stretched) to match the dimensions of the Bitmap object.
4	Size to Fit—the Bitmap object is resized to match the dimensions of the bitmap.

The default value of Layout is 4 – Size to Fit.

LeadingString

LeadingString specifies the non-editable string that appears before the numeric value in the edit window. (For example, the leading string could be set to "$" to make the numeric control appear like a currency control.)

The syntax for the LeadingString property consists of:

```
object.LeadingString [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. The non-editable string that appears before the numeric value in the edit window.

The default value for LeadingString is an empty string (a string of zero length).

ListItem

To add items to an existing list, use the AddString method. To remove items, use the DeleteString method.

ListItem sets or retrieves the list items in a combo box or list box. This property does not give you the selected items in the list box, but sets or retrieves the items that appear within the list.

The syntax of the ListItem property consists of:

```
object.ListItem [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression specifying the items in a list box, or in the list box section of a combo box. Items are separated by the NewLine character.

LocalDecls

LocalDecls stores all of the custom sub and function methods, constants and variables that you can create. Anything that is written here can only be seen by the current View.

LocalVariables

The LocalVariables property allows you to declare variables attached to the Frame that may be used by all associated views. Clicking on LocalVariables in the Property Sheet will open the VBS Mini-Editor, in which you will be able to declare Boolean, Double, DWORD, Long and String type variables. The LocalVariables property gives you the ability to create your own private variables within a frame/view. Local variables are prefixed with an asterisk (*) character within a property list.

Locked

Locked specifies whether a control can be edited through mouse or keyboard input.

A Locked control that is Enabled can still initiate events, receive the focus and perform operations that do not modify the value such as drag operations and copy to clipboard.

The syntax of the Locked property consists of:

```
object.Locked [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the control can be edited.

Possible values for Boolean are:

True	You cannot edit the value.
False	You can edit the value.

The default value for Locked is False (the control can be edited).

LongDate

LongDate specifies the current date using the system-defined long date format. LongDate is a read-only property.

The syntax of the LongDate property consists of:

```
object.LongDate [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that specifies the current date in long date format.

Mask

The MaxLength property is ignored when Mask is set, as Mask itself sets the maximum length.

Mask specifies restrictions on the data that can be typed into or stored in a control. For example, a window containing a phone number may specify that only numeric values up to a certain length may be entered.

See the Masked Edit object for a listing of applicable mask characters.

The syntax of the Mask property consists of:

```
object.Mask [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression specifying restrictions on the data that can be typed into or stored in the control.

The default value for Mask is an empty string (a string of zero length).

MaximizeBox

MaximizeBox indicates whether Windows Maximize form option is enabled. The possible values for MaximizeBox are Yes or No.

Maximizer Field

This property defines which Maximizer Field populates the control. The options for this property depend on which type of object is currently selected.

Maximum

Maximum specifies the maximum value that the user can type into the control.

Maximum is used only if the EnableMaximum property is True.

The syntax of the Maximum property consists of:

```
object.Maximum [ = Double ]
```

object	The required term. Insert the name of a valid object.
Double	The optional term. The maximum value that can be entered into the control.

The default value of Maximum is 100.

MaxLength

MaxLength specifies the maximum number of characters you can type into a control. Setting the MaxLength property to zero puts no limit on the number of characters other than that set by memory

constraints. MaxLength is ignored when the Mask property is set, as the Mask itself sets the maximum length.

The syntax of the MaxLength property consists of:

```
object.MaxLength [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. Insert an integer value limiting the allowed number of characters.

Changing the value of MaxLength restricts only the amount of text that you can type into a control. It does not affect text already in a control, nor the length of the text copied to the edit control by setting the Text property.

If you use the Text property to enter text that exceeds the length specified by MaxLength, you can delete any of that text within the control. However, you will not be able to replace existing text with an amount of text that exceeds the limit set by MaxLength.

The default value for MaxLength is zero (no limit is set).

MinimizeBox

MinimizeBox indicates whether Windows Minimize form option is enabled. The possible values for MinimizeBox are Yes or No.

Minimum

Minimum specifies the minimum value that the user can type into the control. Minimum is used only if the EnableMinimum property is True.

The syntax of the Minimum property consists of:

```
object.Minimum [ = Double ]
```

object	The required term. Insert the name of a valid object.
Double	The optional term. The minimum value that can be entered into the control.

The default value of Minimum is zero (0).

Minute

Minute specifies the minute of the hour.

The syntax of the Minute property consists of:

```
object.Minute [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The minute of the hour.

Minute can take a value in the range 0 to 59.

Month

Month specifies the month of the year. Month is a read-only property.

The syntax of the Month property consists of:

```
object.Month [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The month of the year.

Month can take a value in the range 1 to 12.

MouseIcon

MouseIcon assigns a mouse icon to a control. The mouse icon of a control is the appearance that the mouse pointer takes on as you move it across that control. (You can also assign an image to the mouse pointer by using the Visual Basic LoadPicture method.)

The MouseIcon property is valid when the MousePointer property is set to 99.

The syntax of the MouseIcon property consists of:

```
object.MouseIcon = LoadPicture( pathname )
```

object	A required term. Insert the name of a valid object.
pathname	A required term. Insert the path and filename of the file containing the icon.

MousePointer

MousePointer specifies the type of pointer displayed when you place the mouse pointer over a particular control. You can use this property to show a change in the function of the mouse when it is used on different controls.

The syntax for the MousePointer property consists of:

```
object.MousePointer [ = trkMousePointer ]
```

object	The required term. Insert the name of a valid object.
trkMousePointer	The optional term. A constant that specifies the shape to be used as the mouse pointer.

Possible values for trkMousePointer are:

0	trkMousePointerDefault—display the standard pointer.
1	trkMousePointerArrow—the pointer is an arrow.
2	trkMousePointerCross—the pointer is a cross-hair.
3	trkMousePointerIBeam—the pointer is an I-beam.
6	trkMousePointerSizeNESW—the pointer is a double arrow pointing northeast–southwest.
7	trkMousePointerSizeNS—the pointer is a double arrow pointing north–south.
8	trkMousePointerSizeNWSE—the pointer is a double arrow pointing northwest–southeast.
9	trkMousePointerSizeWE—the pointer is a double arrow pointing east–west.
10	trkMousePointerUpArrow—the pointer is an Up arrow.
11	trkMousePointerHourglass—the pointer is an hourglass.

12	trkMousePointerNoDrop—the pointer is the "Not" symbol—a circle bisected by a diagonal line. It identifies the control being dragged as not a valid drop target.
13	trkMousePointerAppStarting—the pointer is an arrow with an hourglass. (It indicates that you have to wait for an existing process to finish.)
14	trkMousePointerHelp—the pointer is an arrow with a question mark.
15	trkMousePointerSizeAll—the pointer is a "size-all" cursor, with arrows pointing north, south, east, and west.
99	trkMousePointerCustom—the pointer is the icon specified by the MouseIcon property.

The default value for MousePointer is 0 (display the standard pointer).

MultiColumn

MultiColumn specifies whether a list box is a multi-column list box that scrolls horizontally.

The ColumnWidth property sets the width of the columns.

The syntax of the MultiColumn property consists of:

```
object.MultiColumn [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the control is a multi-column list box.

The possible values of Boolean are:

True	The list box is multi-column.
False	The list box is not multi-column.

The default value for MultiColumn is False (there is only one column).

MultiLine

MultiLine specifies whether a control can accept and display multiple lines of text.

The syntax of the MultiLine property consists of:

```
object.MultiLine [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the control supports more than one line of text.

Possible values for Boolean are:

True	The text is displayed across multiple lines.
False	The text is not displayed across multiple lines.

The default value for MultiLine is True (the text is displayed across multiple lines).

NoIntegralHeight

NoIntegralHeight specifies whether the size of a list box is exactly the size specified by an application when it created the list box.

Usually, Windows sizes a list box so that the list box does not display partial items.

The syntax of the NoIntegralHeight property consists of:

```
object.NoIntegralHeight [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the size of the list box is that specified at creation.

The possible values of Boolean are:

True	The size of the list box is as specified.
False	The size of the list box has been modified so as not to display partial items.

The default value for NoIntegralHeight is False (partial items are not displayed).

NonCurDatesTextColor

The NonCurDatesTextColor property defines the color of dates that belong to the previous or next month in the DateTimePicker control's calendar. The syntax of the ColorSaturday property consists of:

```
object.NonCurDatesTextColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color of the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

NumberFont

NumberFont specifies the font used for clock text in the Clock control.

The syntax for the NumberFont property consists of:

```
object.NumberFont [ = Font ]
```

object	The required term. Insert the name of a valid object.
Font	The optional term. The font object used for clock text.

The default value for NumberFont is the font Arial 8.

NumericType

NumericType specifies the type of value that can be stored in a control.

The syntax of the NumericType property consists of:

```
object.NumericType [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer constant that specifies the kind of value that can be stored.

Possible values for Integer are:

0	Short—the value is a short integer.
1	Long—the value is a long integer.
2	Float—the value is a floating-point value.
3	Double—the value is a double-precision floating-point value.

Use this type of limit to ensure that an overflow does not occur (for example, setting NumericType to Short limits the value to integers between -32,768 and 32,767).

It also allows you to enable or disable the floating-point that can be typed in (for example, if NumericType is set to Short, no decimal point can be typed in).

The default value for NumericType is 0 (the value is a short integer).

Orientation

The Orientation control defines the Spinner control's button orientation. Possible values for Orientation are as follows:

0	Horizontal
1	Vertical

The default for Orientation is Vertical.

PenStyle

The PenStyle property defines the appearance of the Line object: whether the line is solid, dashed, dotted, or a combination of dashed and dotted. Possible values for PenStyle are as follows:

0	Solid
1	Dash
2	Dot
3	Dash Dot
4	Dash Dot Dot

The default value for PenStyle is Solid.

Picture

Picture specifies an image that is to be displayed in a control.

The syntax of the Picture property consists of:

```
object.Picture [ = Picture ]
```

object	The required term. Insert the name of a valid object.
Picture	The optional term. It specifies a file that contains the image to be displayed on the control.

PictureAlignment

Setting the PictureSizeMode property to fill a Picture control completely may override the setting of the PictureAlignment property.

PictureAlignment specifies the positioning of the image in a Bitmap control.

The syntax of the PictureAlignment property consists of:

```
object.PictureAlignment [ = trkPictureAlignment ]
```

object	The required term. Insert the name of a valid object.
trkPictureAlignment	The optional term. A constant that specifies the alignment of the picture within the control.

Possible values for the constant trkPictureAlignment are:

0	trkPictureAlignmentTopLeft—aligned to the top left of the control.
1	trkPictureAlignmentTopCenter—aligned to the top center of the control.
2	trkPictureAlignmentTopRight—aligned to the top right of the control.
3	trkPictureAlignmentCenterLeft—aligned to the center left of the control.
4	trkPictureAlignmentCenter—aligned in the center of the control.
5	trkPictureAlignmentCenterRight—aligned to the center right of the control.
6	trkPictureAlignmentBottomLeft—aligned to the bottom left of the control.
7	trkPictureAlignmentBottomCenter—aligned to the bottom center of the control.
8	trkPictureAlignmentBottomRight—aligned to the bottom right of the control.

The PictureAlignment property identifies which points on the image and the Picture control coincide.

For example, trkPictureAlignmentCenter means that the centers of the two objects coincide. Similarly, trkPictureAlignmentTopLeft means that the top left corner of both objects coincide; and trkPictureAlignmentBottomRight means that the bottom right corners coincide.

In the case of a tiled image (where the image is smaller than the size of the Picture control and multiple copies of the same image are used to fill the remaining area), PictureAlignment specifies the placement of the first copy of the image. Additional copies are then placed to line up vertically and horizontally with the first copy.

The default value for PictureAlignment is 0 (top left).

PictureSizeMode

PictureSizeMode specifies how to display the image on a Bitmap object.

The syntax of the PictureSizeMode property consists of:

```
object.PictureSizeMode [ = trkPictureSizeMode ]
```

object	The required term. Insert the name of a valid object.
trkPictureSizeMode	The optional term. A constant that specifies how to re-size the image if it and the Picture control are not the same size.

Possible values for trkPictureSizeMode are:

0	trkPictureSizeModeClip—the image is cropped if it is larger than the control.
1	trkPictureSizeModeStretch—the image is stretched and distorted to fill the Picture control completely. Note that if you choose this value, you will override the setting of the PictureAlignment property.
3	trkPictureSizeModeZoom—the undistorted image is enlarged until either its horizontal or vertical edges coincide with the corresponding edges of the Picture control.

The default value for PictureSizeMode is 0 (the image is cropped if it is larger than the control).

PictureTiling

The way in which an image is tiled across the control also depends on the values you have chosen for the control's PictureAlignment and PictureSizeMode properties.

PictureTiling specifies whether an image is tiled across a Bitmap object.

The syntax of the PictureTiling property consists of:

```
object.PictureTiling [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether an image is repeated to fill the Bitmap object.

Possible values for Boolean are:

True	The image is tiled across the Picture control.
False	The image is not tiled across the Picture control.

The default value for PictureTiling is False (the image is not tiled across the control).

PrintScale

PrintScale is a property of TheFrame/TheView that changes the scaling of the frame when it is printed. The default value of PrintScale is 1.0.

RadioGroupValue

RadioGroupValue sets or retrieves the value of a radio button group.

Setting this property selects the radio button in the group that matches this value. Retrieving this value returns the value of the currently selected radio button within the group. If setting the property to a value that does not match any of the radio buttons DataSourceValue property, all radio buttons will be unchecked.

The syntax of the RadioGroupValue property consists of:

```
object.RadioGroupValue [ = Variant ]
```

object	The required term. Insert the name of a valid object.
Variant	The optional term. The value of the radio button group.

RadioGroupValue is only valid at run-time.

ReadOnly

The ReadOnly property determines whether the control is in "read-only" mode, which sets the control to display-only. The default value of ReadOnly is 0 – No.

ReportLock

ReportLock, which is a property of TheFrame/TheView, defines whether the frame is a static report view and not updated by new

data. ReportLock is a boolean property with the default value of 0 - No.

RomanNumerals

RomanNumerals specifies whether Roman numerals or digits are used in the clock text of the Clock control.

The syntax of the RomanNumerals property consists of:

```
object.RomanNumerals [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression specifying whether Roman numerals or digits are used in clock text.

The possible values for Boolean are:

True	Roman numerals are used.
False	Digits are used.

The default value for RomanNumerals is False (use digits for the clock text).

SaveSelection

SaveSelection specifies whether previously selected text is highlighted again when a control regains the focus.

The syntax of the SaveSelection property consists of:

```
object.SaveSelection [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the previously selected text is highlighted again when a control regains the focus.

Possible values for Boolean are:

True	Previously selected text is highlighted again.
False	Previously selected text is not highlighted.

The default value for SaveSelection is True (previously selected text is highlighted again).

ScrollBars

ScrollBars specifies whether a Text control can have vertical and/or horizontal scroll bars. Even if scroll bars are enabled, they will not appear unless the contents of the control take up more space than has been allocated to the control.

The presence or absence of scroll bars may also depend on the values you have chosen for the MultiLine property of the control.

The syntax for the ScrollBars property consists of:

```
object.ScrollBars [ = trkScrollBars ]
```

object	The required term. Insert the name of a valid object.
trkScrollBars	The optional term. A constant that specifies whether and where scroll bars should be displayed on the control.

Possible values for trkScrollBars are:

0	trkScrollBarsNone—scroll bars are disabled. No scroll bars are displayed.
1	trkScrollBarsHorizontal—horizontal scroll bar may be displayed below the control.
2	trkScrollBarsVertical—vertical scroll bar may be displayed to the right of the control.
3	trkScrollBarsBoth—both horizontal and vertical scroll bars may be displayed.

A vertical scroll bar will appear on a control only if trkScrollBars has the values 2 or 3, and the number of lines in the contents exceeds the number allocated to the control, and the control has enough room to display the scroll bar.

A horizontal scroll bar will appear on a control only if trkScrollBars has the values 1 or 3, and the width of the contents exceeds the width allocated to the control, and the control has enough room to display the scroll bar.

The default value of ScrollBars is 0 (no scroll bars are displayed).

ScrollHeight

ScrollHeight defines the size in pixels up to which scroll bars will be displayed on the form. If the form is resized to a height that is greater than the ScrollHeight value, then the scroll bar(s) will not be displayed. For this property to function, the ScrollBars property must have a non-zero value.

ScrollWidth

ScrollWidth defines the size in pixels up to which scroll bars will be displayed on the form. If the form is resized to a width that is greater than the ScrollWidth value, then the scroll bar(s) will not be displayed. For this property to function, the ScrollBars property must have a non-zero value.

Second

Second specifies the second of the minute.

The syntax of the Second property consists of:

```
object.Second [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. The second of the minute.

Second can take a value in the range 0 to 59.

SelAlignment

SelAlignment specifies the alignment of a paragraph in a Rich Text control.

The syntax of the SelAlignment property consists of:

```
object.SelAlignment [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional value. An integer that specifies the paragraph alignment in the control.

Possible values for SelAlignment are:

Null	You have selected more than one paragraph having different alignments.
0	The paragraph is left-aligned.
1	The paragraph is right-aligned.
2	The paragraph is centered.

The default value for SelAlignment is zero (the paragraph is left-aligned).

SelBold

SelBold sets the font style of the currently selected text to bold.

The syntax of the SelBold property consists of:

```
object.SelBold [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies that the selected text is bold.

The possible values for Boolean are:

True	All the text selected is bold.
False	None of the text selected is bold.

SelBold is set to null if the selected text contains a mixture of bold and non-bold text.

The default value for SelBold is False (the text is not bold).

SelBullet

SelBullet specifies if the selected paragraph in the control is a "bullet paragraph".

The syntax of the SelBullet property consists of:

```
object.SelBullet [ = Boolean ]
```

object	The required term. Insert the name of valid control.
Boolean	The optional term. A Boolean expression that specifies if the selected paragraph is in "bullet" style.

The possible values for Boolean are:

True	The paragraph is a bullet paragraphs.
False	The paragraph is not a bullet paragraph.

SelBullet is set to null if the selected text contains a mixture of bulleted and non-bulleted text.

The default value for SelBullet is False (the paragraph is non-bulleted).

SelCharOffset

SelCharOffset specifies if the text in the control appears normally positioned on a line, or is superscripted or subscripted.

The syntax of the SelCharOffset property consist of:

```
object.SelCharOffset [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer that specifies whether the selected text is normally aligned, superscripted or subscripted.

Possible values for Integer are:

0	The characters all appear as normally aligned text.
Positive integer	The characters are superscripted above the normal baseline by the number of twips specified.
Negative integer	The characters are subscripted below the normal baseline by the number of twips specified.

SelCharOffset is set to null if the selected text contains a mixture of normal, superscripted and subscripted text.

The default value for SelCharOffset is zero (the text is normally aligned).

SelColor

SelColor specifies the color of text in a Rich Text control.

The syntax of the SelColor property consists of:

```
object.SelColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the color of text in the control.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

SelColor is set to null if the selected text contains a mixture of different colored text.

SelectedDateFont

SelectedDateFont specifies the font used for the currently selected date.

The syntax of the SelectedDateFont property consists of:

```
object.SelectedDateFont [ = Font ]
```

object	The required term. Insert the name of a valid object.
Font	The optional term. The font object used for the selected date.

The default value for SelectedDateFont is the font Arial 8.

SelectedHighlightBackColor

SelectedHighlightBackColor specifies the background color for the currently selected date if this date is also highlighted.

The syntax of the SelectedHighlightBackColor property consists of:

```
object.SelectedHighlightBackColor [ = Color ]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the background color for highlighted dates.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

SelectionType

SelectionType specifies how items in a list box may be selected.

The syntax for the SelectionType property consists of:

```
object.SelectionType [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer constant specifying how list items may be selected.

Possible values for Integer are:

0	Single—one item only may be selected at a time.
1	Multiple—string selection is toggled every time the user clicks or double-clicks the string.
2	Extended—the user can select multiple items using the Shift key and the mouse or special key combinations.

The default value for SelectionType is 0 (select one item only at a time).

SelFontName

SelFontName specifies the font used to display the currently selected text.

The syntax of the SelFontName property consists of:

```
object.SelFontName [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that identifies a valid font.

SelFontName is set to null if the selected text contains different fonts.

SelFontSize

SelFontSize sets the font size (in points) of selected text in a Rich Text control.

```
object.SelFontSize [ = Integer ]
```

The syntax of the SelFontSize property consists of:

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value that specifies the size of currently selected text in the Rich text control.

SelFontSize property is set to null if the selected text contains different font sizes.

SelHangingIndent

SelHangingIndent specifies the margin settings for paragraphs in the Rich text control with a hanging indent.

A hanging indent is the distance between the left edge of the first line of text in a paragraph and the left edge of the second and later lines of text in the same paragraph.

The syntax of the SelHangingIndent property consists of:

```
object.SelHangingIndent [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer that determines the amount of indent in a paragraph. The size of the indent is scaled to the size of the form that contains the Rich Text control.

SelHangingIndent is set to zero (0) if the selected text contains paragraphs with different indent settings.

SelIndent

SelIndent specifies the margin settings for paragraphs in a Rich Text control with a left indent.

An indent specifies the distance between the left edge of the control and the left edge of the selected paragraph.

The syntax of the SelIndent property consists of:

```
object.SelIndent [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer that determines the amount of indent in a paragraph. The size of the indent is scaled to the size of the form that contains the Rich Text control.

SelIndent is set to zero (0) if the selected text contains paragraphs with different indent settings.

SelItalic

SelItalic sets the font style of the currently selected text to italic.

The syntax of the SelItalic property consists of:

```
object.SelItalic [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies that the selected text is italic.

The possible values for Boolean are:

True	All the text selected is italic.
False	None of the text selected is italic.

SelItalic is set to null if the selected text contains a mixture of italic and non-italic text.

The default value for SelItalic is False (the text is not italic).

SelLength

SelLength specifies the number of characters selected in a Rich Text control. It is used in combination with the SelStart property to select text in a control. (Changing the value of a control's SelStart property will set SelLength to zero.)

The syntax of the SelLength property consists of:

```
object.SelLength [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. Insert a numeric expression specifying the number of characters selected.

The default value for SelLength is zero (no text is currently selected).

Allowed values for SelLength range from zero to n, where n is the total number of characters in the control. Entering a value that is less than zero will cause an error; entering a value greater than n will set selLength to n.

SelRightIndent

SelRightIndent specifies the margin settings for paragraphs in a Rich Text control with a right indent.

A right indent specifies the distance between the right edge of the selected paragraph and the right edge of the control.

The syntax of the SelRightIndent property consists of:

```
object.SelRightIndent [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer that determines the amount of indent in a paragraph. The size of the indent is scaled to the size of the form that contains the Rich Text control.

SelRightIndent is set to zero (0) if the selected text contains paragraphs with different indent settings.

SelStart

SelStart specifies the starting point of selected text. It is used in combination with the SelLength property to select text in a control. If no text is currently selected (SelLength equals zero), SelStart specifies the insertion point in the control.

Changing the value of SelStart cancels any existing selection in the control (setting SelLength to zero), and places an insertion point in the text.

The syntax of the SelStart property consists of:

```
object.SelStart [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. Insert a numeric expression specifying the starting point of selected text.

The default value for SelStart is zero.

Allowed values for SelStart range from zero to n, where n is the total number of characters in the control. Entering a value that is less than zero will cause an error; entering a value greater than n will set SelStart to n.

SelStrikeThrough

SelStrikethrough sets the font style of the currently selected text to strikethrough.

The syntax of the SelStrikethrough property consists of:

```
object.SelStrikethrough [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies that the selected text is strikethrough.

The possible values for Boolean are:

True	All the text selected is strikethrough.
False	None of the text selected is strikethrough.

SelStrikethrough is set to null if the selected text contains a mixture of strikethrough and non-strikethrough text.

The default value for SelStrikethrough is False (the text is not strikethrough).

SelTabCount

SelTabCount specifies the number of tabs in a Rich Text control.

The syntax of the SelTabCount property consists of:

```
object.SelTabCount [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer that determines the number of tab positions in a selected paragraph.

SelTabs

SelTabs specifies the absolute tab positions of text in a Rich Text control.

The syntax of the SelTabs property consists of:

```
object.SelTabs(Index) [ = Integer ]
```

object	A required term. Insert the name of a valid object.
Index	A required term. An integer that identifies a specific tab. Index takes a value in the range of 0 to (SelTabCount-1). The first tab location has an index of zero (0). The last tab location has an index equal to SelTabCount minus 1.
Integer	An optional term. An integer that specifies the location of the designated tab. The units used to express tab positions are scaled to the size of the form containing the Rich Text control.

SelText

SelText specifies the selected text within a control.

The syntax of the SelText property consists of:

```
object.SelText [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. Insert a string expression containing the selected text.

If no characters within the control are selected, the SelText property returns a string of zero length.

SelUnderline

SelUnderline sets the font style of the currently selected text to underlined.

```
object.SelUnderline [ = Boolean ]
```

The syntax of the SelUnderline property consists of:

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies that the selected text is underlined.

The possible values for Boolean are:

True	All the text selected is underlined.
False	None of the text selected is underlined.

SelUnderline is set to null if the selected text contains a mixture of underlined and non-underlined text.

The default value for SelUnderline is False (the text is not underlined).

ShadowStyle

The ShadowStyle property should not be confused with the BorderStyle property, which defines a "3-D" appearance to the Text object's border.

The ShadowStyle property defines whether or not the Text control has a drop shadow, which can make the Text control appear to be raised off the form. The possible values of ShadowStyle are as follows:

0	None—the text box has no drop shadow.
1	Top/Left—the drop shadow appears on the top left edges of the text object.
2	Bottom/Right—the drop shadow appears on the bottom left edges of the text object.

The default value for ShadowStyle is None.

ShortDate

ShortDate specifies the current date using the system-defined short date format. ShortDate is a read-only property.

The syntax of the ShortDate property consists of:

```
object.ShortDate [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that specifies the current date in short date format.

ShowAdjacentMonths

ShowAdjacentMonths specifies whether dates from the previous and following months are displayed.

The syntax of the ShowAdjacentMonths property consists of:

```
object.ShowAdjacentMonths [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether dates from the previous and following months are displayed.

The possible values for Boolean are:

True	Dates from the previous and following months are shown.
False	Dates from the previous and following months are not shown.

The default value for ShowAdjacentMonths is True (the previous and following months are shown).

ShowAllHighlightedDates

ShowAllHighlightedDates specifies that all selected dates should be displayed in the colors specified by the HighlightColor and HighlightBackColor properties.

The syntax of the ShowAllHighlightedDates property consists of:

```
object.ShowAllHighlightedDates [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether highlighted dates are shown in the colors specified by the HighlightColor and HighlightBackColor properties.

The possible values for Boolean are:

True	Highlighted dates are shown using the HighlightColor and HighlightBackColor properties.
False	Highlighted dates are shown as normal.

The default value for ShowAllHighlightedDates is True (the colors used are those specified by the HighlightColor and HighlightBackColor properties).

ShowAMPM

ShowAMPM specifies whether the Clock control displays the AM/PM indicator.

The syntax of the ShowAMPM property consists of:

```
object.ShowAMPM [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression specifying whether the AM/PM indicator is shown on the clock face.

The possible values for Boolean are:

True	Show the AM/PM indicator.
False	Hide the AM/PM indicator.

The default value for ShowAMPM is True (the indicator is shown).

ShowButton

ShowButton specifies whether a Dial, Launch, Sendmail or spin button is displayed on the appropriate control. Even when the button is not displayed, the associated function in the Email, Launch and Phone controls can still be called.

The syntax for the ShowButton property consists of:

```
object.ShowButton [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the button is displayed.

The possible values for Boolean are:

True	The button is displayed.
False	The button is not displayed.

The default value for ShowButton is True (the button is displayed).

ShowButtons

ShowButtons specifies whether the Month and Year navigation buttons are visible on the control.

The syntax of the ShowButtons property consists of:

```
object.ShowButtons [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression specifying whether the Month and Year navigation buttons are visible.

The possible values of Boolean are:

True	The Month and Year navigation buttons are visible.
False	The buttons are not visible.

The default value for ShowButtons is True (the Month and Year navigation buttons are visible).

ShowCalendarButton

ShowCalendarButton specifies whether the Calendar button is displayed on the DateTimePicker control.

The syntax for the ShowCalendarButton property consists of:

```
object.ShowCalendarButton [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the button is displayed.

The possible values for Boolean are:

True	The button is displayed.
False	The button is not displayed.

The default value for ShowCalendarButton is True (the button is displayed).

ShowClockButton

ShowClockButton specifies whether the Clock button is displayed on the DateTimePicker control.

The syntax for the ShowClockButton property consists of:

```
object.ShowClockButton [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the button is displayed.

The possible values for Boolean are:

True	The button is displayed.
False	The button is not displayed.

The default value for ShowClockButton is True (the button is displayed).

ShowDayOfWeek

ShowDayOfWeek specifies whether the day of the week is shown when the DateTimePicker control is used in Date mode

The syntax of the ShowDayOfWeek property consists of:

```
object.ShowDayOfWeek [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the day of the week is displayed.

The possible values for Boolean are:

True	The day of the week is displayed.
False	The day of the week is not displayed.

The default value of ShowDayOfWeek is True (the day of the week is displayed).

ShowOnlyHour

ShowOnlyHour specifies whether the Clock control displays only the Hour hand, or whether the control displays both Hour and Minute hands.

The syntax of the ShowOnlyHour property consists of:

```
object.ShowOnlyHour [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression specifying whether the hour hand only is shown, or whether both Hour and Minute hands are shown.

The possible values for Boolean are:

True	Show the Hour hand only.
False	Show both hands.

The default value for ShowOnlyHour is False (both Hour and Minute hands are shown).

ShowSeconds

ShowSeconds specifies whether seconds are shown when the DateTimePicker control is used in Time mode

The syntax of the ShowSeconds property consists of:

```
object.ShowSeconds [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether seconds are displayed.

The possible values for Boolean are:

True	Seconds are displayed.
False	Seconds are not displayed.

The default value of ShowSeconds is True (seconds are displayed).

ShowSelectedDate

ShowSelectedDate specifies that the currently selected date is shown using the font set in the SelectedDateFont property.

The syntax of the ShowSelectedDate property consists of:

```
object.ShowSelectedDate [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression specifying whether the selected date is displayed in the font set in the SelectedDateFont property.

The possible values of Boolean are:

True	The currently selected date is shown using the SelectedDateFont property.
False	The currently selected date is shown like a non-selected date.

The default value for ShowSelectedDate is True (the value of the SelectedDateFont property is used).

ShowWeekNumbers

ShowWeekNumbers specifies whether week numbers are visible on the Calendar control.

The syntax of the ShowWeekNumbers property consists of:

```
object.ShowWeekNumbers [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether week numbers are visible.

The possible values for Boolean are:

True	Week numbers are visible.
False	Week numbers are not visible.

The default value for ShowWeekNumbers is True (week numbers are visible).

Sort

Sort specifies whether strings entered into a list box, or into the list-box section of a combo box, are automatically sorted.

The syntax of the Sort property consists of:

```
object.Sort [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the entered strings are automatically sorted

Possible values of Boolean are:

True	Strings will be automatically sorted.
False	Strings will not be automatically sorted.

The default value for Sort is True (strings will be automatically sorted).

TabWidth

TabWidth sets the distance between tab-stops in a list box.

The syntax for the TabWidth property consists of:

```
object.TabWidth [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer constant specifying how far apart tab stops are set in the list box.

Tab-stop distances are measured in dialog units. The default value for TabWidth is 2 units.

Text

Text specifies the text in the editable area of a control. Text is valid only at run-time. The ForeColor property determines the color of the text.

```
object.Text [ = String ]
```

The syntax of the Text property consists of:

object	The required term. Insert the name of a valid object.
String	The optional term. Insert a string expression specifying the text to be set or returned.

For a list box, the value of Text must match an existing list entry. Specifying a value that does not match an existing list entry causes any previously selected items to be deselected.

The default value for Text is an empty string (a string of zero length).

TextLimit

The TextLimit property defines the maximum number of characters which may be entered into an edit control at run time. The default value of zero allows an unlimited number of characters.

TickFrequency

The TickFrequency property defines the number tick intervals to display on a slider control. The syntax of TickFrequency consists of:

```
object.TickFrequency [ = Long ]
```

object	Insert the name of a valid Slider object.
Long	Enter the number of tick intervals to display (2-100).

The default value for TickFrequency is 10.

TickStyle

The TickStyle property for the Slider object defines where the ticks appear on the control. The syntax of TickStyle consists of:

```
object.TickStyle [ = Integer ]
```

object	Insert the name of a valid Slider object.
Integer	Insert an integer value that defines the location of the tick marks.

The possible values of *Integer* are as follows:

0	Both—the tick marks appear on both the top and bottom of the slider if it is horizontal or both left and right if it is vertical.
1	Top/Left—the tick marks appear on the top of the slider if it is horizontal or on the left if the slider is vertical.
2	Bottom/Right—the tick marks appear on the bottom of the slider if it is horizontal or on the right if the slider is vertical.

The default value for TickStyle is 0 – Both.

Title

Title defines the text title that appears on the control. The syntax of the Title property consists of:

```
object.Text [ = String]
```

object	Insert the name of a valid object.
String	A string containing the text of the title.

TitleBackColor

TitleBackColor defines the background color for the DateTimePicker control. The syntax for the TitleBackColor property consists of:

```
Object.TitleBackColor [ = Color]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the title's background color.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

TitleFormat

TitleFormat specifies the date format used for a heading in the DateTimePicker control.

The syntax for the TitleFormat property consists of:

```
object.TitleFormat [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression specifying the date format used.

The default value for TitleFormat is "MMM yyyy", where MMM is the three-character abbreviation for the month, and yyyy is the four-digit value for the year.

TitleTextColor

TitleTextColor defines the foreground color for the DateTimePicker control. The syntax for the TitleTextColor property consists of:

```
Object.TitleTextColor [ = Color]
```

object	The required term. Insert the name of a valid object.
Color	The optional term. Insert an integer value that determines the title's text color.

The possible values for Color include any integer that represents a valid color. You can also specify a color by using the RGB function, which describes colors as a mixture of Red, Green, and Blue (*e.g.*, RGB (255,0,0), which would produce the color red).

ToolBar

ToolBar specifies whether the toolbar appears with its associated Rich Text control. The AutoHideToolBar property is ignored if ToolBar is set to False (see below).

The syntax of the ToolBar property consists of:

```
object.ToolBar [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the toolbar appears with its control.

The possible values for Boolean are:

True	The toolbar appears.
False	The toolbar is hidden.

The default value for the ToolBar property is True (the toolbar appears with its associated control).

TrailingString

TrailingString specifies the non-editable string that appears after the numeric value in the edit window.

```
object.TrailingString [ = String ]
```

The syntax for the TrailingString property consists of:

object	The required term. Insert the name of a valid object.
String	The optional term. The non-editable string that appears after the numeric value in the edit window.

The default value for TrailingString is an empty string (a string of zero length).

TrkDataCard

TrkDataCard specifies the card name for Tracker data awareness. Tracker data awareness is in addition to Microsoft data awareness available under Visual Basic.

The syntax of the TrkDataCard property consists of:

```
object.TrkDataCard [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. Insert a string expression specifying the card name for Tracker data awareness.

The default value for TrkDataCard is an empty string (a string of zero length).

Type

Type in the Calendar control specifies whether the Calendar appears in Western or Japanese style. Type in the DateTimePicker control specifies whether the control is a Date control, a Time control, or both.

The syntax of the Type property consists of:

```
object.Type [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer value specifying the type of control displayed in the Calendar and DateTimePicker controls.

The possible values for Integer in the Calendar control are:

0	Western style
1	Japanese style

The possible values for Integer in the DateTimePicker control are:

0	The control is a Date control
1	The control is a Time control
2	The control is both a Date and Time control Default Value The default value for Type in both controls is zero (0).

UncheckedValue

UncheckedValue specifies whether a check box control is left blank ("unchecked"). The properties CheckedValue and UncheckedValue together can be used as a Boolean pair. The property defaults to 'False' but can be set to other values such as 'Paid' and 'Unpaid' etc.

The strings set in the CheckedValue and UncheckedValue properties are returned as a variant in the value property, depending upon the checked state. Since Value is a variant, it can be used as any type that the variant conversion functions support, Hence the with the default strings as 'True' and 'False', the Value property can be used as a Boolean value.

The syntax of the UncheckedValue property consists of:

```
object.UncheckedValue [ = String ]
```

object	The required term. Insert the name of a valid object.
String	The optional term. A string expression that specifies whether the check box is "unchecked".

The default value for UncheckedValue is False (the check box is checked).

UpDown

UpDown defines the functionality of the DateTimePicker control. Possible vales for the UpDown property are as follows:

0	No—a single button is displayed, which opens a monthly calendar when clicked.
1	Yes—an up/down button pair is displayed, which increments or decrements the value in the control.

The default value for UpDown is No.

UseColors

The UseColors property determines whether the control uses the color values set in the ForeColor and BackColor properties. The syntax of the UseColors property consists of:

```
object.UseColors [ = Boolean ]
```

object	Insert the name of a valid object.
Boolean	A Boolean expression that specifies whether the ForeColor or BackColor properties are used.

The default value for UseColors is 1 – Yes.

UseTabStops

UseTabStops allows a list box to recognize and expand tab characters when drawing its strings.

The syntax of the UseTabStops property consists of:

```
object.UseTabStops [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the list box can recognize and expand tab characters.

The possible values of Boolean are:

True	The list box can recognize and expand tab characters.
False	The list box cannot recognize and expand tab characters.

The default value for UseTabStops is False (the list box cannot recognize and expand tab characters).

Value

Value specifies the content of a control. The Value property is valid only at run-time.

The syntax of the Value property consists of:

```
object.Value [ = Variant ]
```

object	The required term. Insert the name of a valid object.
Variant	The optional term. It is the content of the control.

ViewLayers

Clicking on the ViewLayers property in the Property Sheet displays the Layers Sheet, which allows the you to control which layers are visible and which are not. This property is only available at design time.

VScroll

VScroll enables or disables the vertical scroll bar.

The syntax of the VScroll property consists of:

```
object.VScroll [ = Boolean ]
```

object	The required term. Insert the name of a valid object.
Boolean	The optional term. A Boolean expression that specifies whether the combo box or list box may have a vertical scroll bar.

Possible values of Boolean are:

True	The vertical scroll bar is enabled.
False	The vertical scroll bar is disabled.

The default value for VScroll is False (the scroll bar is disabled).

WeekNumberFont

WeekNumberFont specifies the font used for week numbers in the Calendar control.

The syntax of the WeekNumberFont property consists of:

```
object.WeekNumberFont [ = Font ]
```

object	The required term. Insert the name of a valid object.
Font	The optional term. The font object used for week numbers.

The default value for WeekNumberFont is the font Arial 8.

WhatsThisHelp

The WhatsThisHelp property defines whether the control-level context Help is activated in the frame. In order for the context Help to work, you must enter the correct Help context ID in each object's HelpContextID property. Possible values for WhatsThisHelp are Yes or No.

Width

The Width property defines the width of TheFrame in pixels. The syntax of the Width property consists of:

```
TheFrame.Width [ = Long ]
```

Long	The width of TheFrame in pixels.

WrapAround

The WrapAround property of the Spinner object determines whether the value displayed "wraps around" when the user increments above the maximum value or decrements below the minimum value.

```
object.WrapAround [ = Boolean ]
```

object	Insert the name of a valid Spinner object.
Integer	A boolean value defining whether or not the spinner value wraps.

The default value of WrapAround is 1 – Yes.

Year

Year specifies the currently selected year.

The syntax of the Year property consists of:

```
object.Year [ = Integer ]
```

object	The required term. Insert the name of a valid object.
Integer	The optional term. An integer that specifies the currently selected year.

Year can a value in the range 1601 to 9999.