Chapter 2
The Toolbox

It seems every new version of Visual FoxPro introduces some new tools. VFP 8 brings three of them, all written in VFP itself. This chapter looks at the Toolbox, which provides a new way to access controls and much more.

The availability of tools is one of the things that distinguishes a productive development environment from a less productive one. VFP has always included a variety of tools (going back to the FoxBase days), and it seems each new version of the product brings more tools to help us work faster and smarter.

The Toolbox (shown in **Figure 1**) is the new preferred tool for adding controls to forms and classes. It combines the simplicity of the Form Controls toolbar with the flexibility of the Component Gallery to provide a tool better than either.

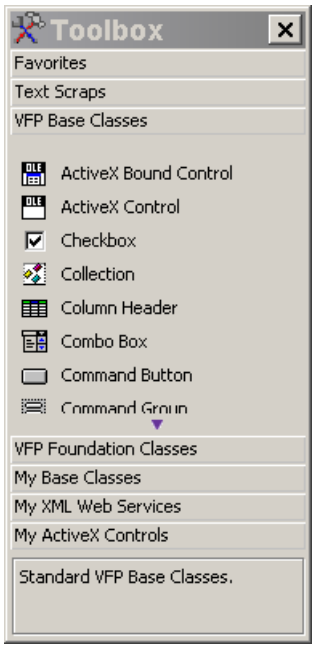

Figure 1. *The Toolbox. The Toolbox offers a marriage of the Form Controls toolbar and the Component Gallery.*

The Toolbox is divided into categories, each represented by a bar in the Toolbox. Each category contains a list of items. There are several types of categories and several types of items (described later in the chapter). By default, there are seven categories: Favorites, Text Scraps, VFP Base Classes, VFP Foundation Classes, My Base Classes, My XML Web Services, and My ActiveX Controls.

Also by default, the VFP Base Classes category contains all the base classes, while My Base Classes contains the first-level subclasses from the _Base library in the FoxPro Foundation Classes. (You can change the contents of My Base Classes, but not VFP Base Classes. See "Adding items to the Toolbox" later in this chapter.) VFP Foundation Classes contains a number of the more useful classes (like the About dialog and the Registry class) from the FoxPro Foundation Classes. The initial contents of the other categories are described later in the chapter.

Opening the Toolbox

The easiest way to start the Toolbox is to click on the icon for it (a crossed hammer and wrench) on the Standard toolbar. You can also open the Toolbox by choosing it from the Tools menu or by issuing the command:

```
DO (_TOOLBOX)
```

The _TOOLBOX system variable points to the Toolbox application, by default, ToolBox.APP in the home directory.

Unlike the Form Controls toolbar, the Toolbox doesn't open automatically when you open the Form Designer or Class Designer. (In fact, in VFP 8, the Form Controls toolbar doesn't open automatically any more unless you set it up to do so; see Chapter 5, "Better Tools.") You might want to write a little program that opens the toolbox, and then opens a specified form or class. A very simple version of such a program might look like this:

```
LPARAMETERS cForm

DO (_TOOLBOX)
IF EMPTY(cForm)
  MODIFY FORM ? nowait
ELSE
  MODIFY FORM (cForm) NOWAIT
ENDIF
```

Once the Toolbox is open, it can be resized both horizontally and vertically. The size and position of the Toolbox are stored in the resource file.

By default, the Toolbox is set to be always on top. You can change that setting from the context menu. Another context menu item lets you turn off the Help text pane at the bottom of the Toolbox, providing more room for the contents.

Working with the Toolbox

You can do a variety of things with the Toolbox, but its principal function is to put controls onto forms or classes. To open a category so you can access the items within it, click on the name of the category. If the items in the category don't fit in the Toolbox's area along with the category bars, up arrow and/or down arrow bars are added at the top and bottom to allow you to scroll within the category. (In Figure 1, there's a down arrow bar for the VFP Base Classes category.) Click on one of the arrow bars to scroll up or down one item. Better, holding the mouse over one of these bars scrolls continuously in the specified direction. (You can set the scroll rate—see "Configuring the Toolbox" later in this chapter.)

When the Form Designer or Class Designer is open, double-clicking on a control in the Toolbox (whether one of the built-in controls, a subclass, or an ActiveX control) drops an instance of that control onto the active form or class. If a container class is selected in the Designer, the new control is added to that container. If not, the new control is placed in the upper left corner of the form or class.

You can also drag a control from the Toolbox and drop it onto a form or class. If you drop onto a container control, the new control is added to the container; otherwise, it's placed directly on the form or class at the point you dropped it. (Dropping onto a container doesn't require the container to be selected.)

If you attempt to add a control to something that can't hold the control, you get an error message "Object class is invalid for this container." For example, the error appears if you attempt to drop anything other than an option button onto an option group. Some classes, such as Exception and Session, are really meant only for programmatic use. Attempting to add one of these to a form or class results in the message "This class has no visual representation and therefore cannot be dropped onto this container."

Some of the controls have additional smarts. If you attempt to add a page class onto a form or class, a pageframe is automatically added to hold it. Similarly, adding a grid column class onto an appropriate container other than a grid creates a one-column grid in the container. Adding a single option button creates an option group.

We expected we'd be able to drag classes and other files from the Toolbox into a project to have them added to the project, but dragging onto a project gives the "no drop" symbol. Since the Component Gallery supports this functionality, we're a little disappointed. Of course, once you use a class in a member of a project, the class library will be added to the project automatically.

The Toolbox is also handy when working with code. Drag an object out of the Toolbox into a code editing window and a line of code instantiating the class is added. For example, if you drag the Splash Screen class from the VFP Foundation Classes category into a code window, this line of code is generated:

```
_splash = NEWOBJECT("_splash", "_DIALOGS.VCX")
```

In fact, this is simply default behavior. You can specify two distinct behaviors for dragging into code windows—see "Determining the behavior of class items" later in this chapter. (One change you may want to make is to include the path for the class library in the generated code. The default code doesn't include it, which isn't a problem if you're building an application, but might be for utility code.)

Working with categories

The Toolbox supports five kinds of categories, as shown in **Table 1**. **Table 2** shows the types for the built-in categories.

Table 1. *Toolbox category types. Each type of category has some unique characteristics.*

Category type	Purpose
General category	Contains classes and other files.
Dynamic folder category	Contains all or a specified subset of the files in a specified folder.
Registered ActiveX controls	Contains ActiveX controls.
Text scraps	Contains fragments of text, which may contain strings to be expanded using text merge.
XML web services	Contains registered XML web services.

Table 2. *Built-in category types. The categories that come with the Toolbox demonstrate most of the types.*

Category	Type
Favorites	Favorites (special type not available for new categories)
Text Scraps	Text scraps
VFP Base Classes	General
VFP Foundation Classes	General
My Base Classes	General
My XML Web Services	XML Web Services
My ActiveX Controls	Registered ActiveX controls

Although each category has a type, you're actually not restricted to putting items of the specified type in the category. Some of the category types provide special behaviors not included in the General category. For example, a Dynamic folder category always includes all the items from the specified folder. However, you can also add other items to the category if you wish. (See "Adding items to the Toolbox" later in this chapter.) You see this behavior at work in the built-in My XML Web Services category. This category (like all XML web services type categories) includes all registered web services. However, it also includes a number of other items, such as Register and Generic Handler.

To add a new category, open the context menu and choose Add Category. The dialog shown in **Figure 2** appears. Specify the name of the new category and choose its type. When you click OK, the new category is added to the Toolbox, but has no contents.

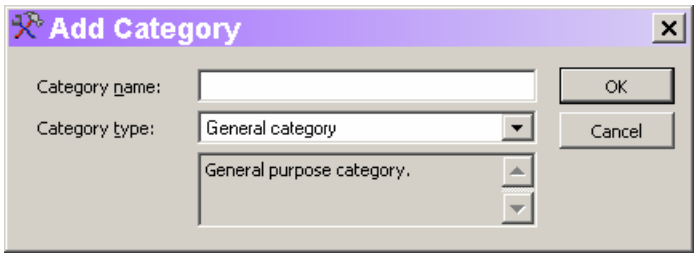

Figure 2. *Adding a new category. When you choose to add a category, this dialog appears. Specify the name and type of the new category.*

To change the settings for a category, right click on the category name in the Toolbox and choose Properties from the context menu. (If the category is expanded, you need to click on the category name before right clicking.) The dialog that appears varies with the category type. **Figure 3** shows the Category Properties dialog for a general category. **Figure 4** shows the dialog for a dynamic folder category. **Table 3** shows the various items that appear in the Category Properties dialog and indicates to which category types they apply.

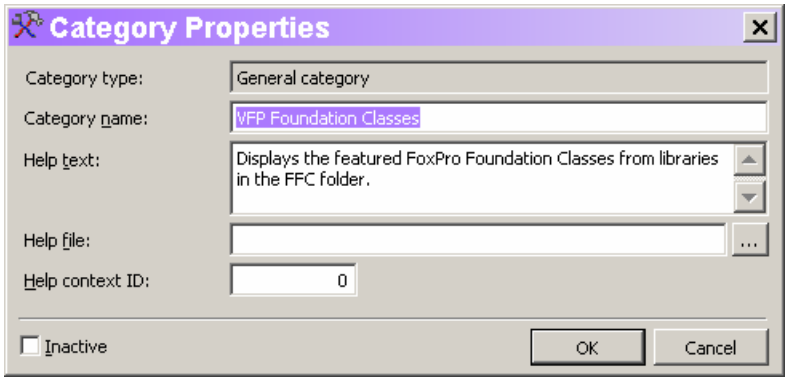

Figure 3. *Changing category settings. For a general category, you can set only a few items.*

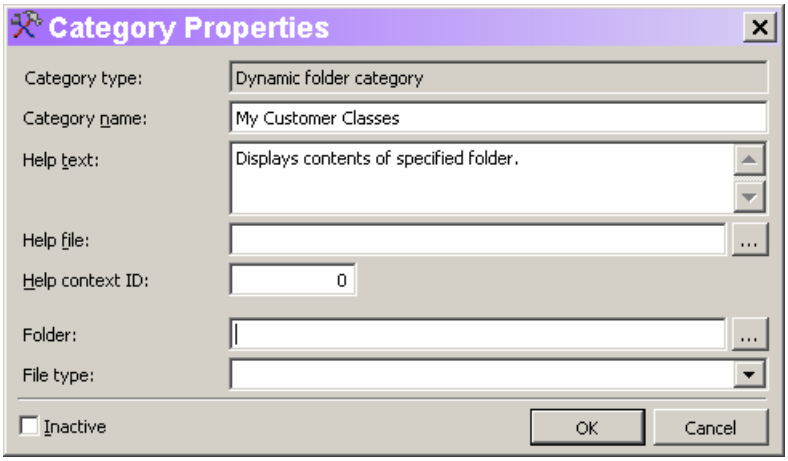

Figure 4. *Setting dynamic folder category properties. When a category is the dynamic folder type, you can also specify the folder and the type(s) of files listed.*

Table 3. *Category properties. The list of properties you can set for a category varies with the category type.*

Property	Applies to	Notes
Category type	All category types	Read-only, set when the category is created.
Category name	All category types	Specifies the name displayed for the category in the Toolbox.
Help text	All category types	Specifies the text shown in the Help pane at the bottom of the Toolbox when the category bar has focus.
Help file	All category types	Specifies a Help file containing Help for this category.
Help context ID	All category types	Specifies the help context id for the topic that should be displayed when Help opens. If this is specified and Help file is empty, it refers to the VFP Help file.
Inactive	All category types	Indicates whether the category is displayed in the toolbox at this time.
Folder	Dynamic folder type	Specifies the folder whose files are displayed for this category.
File type	Dynamic folder type	Specifies a set of file extensions. Only files in the specified folder having those extensions are included in the category. One reason to limit the extensions is so you see only the "main" file of items stored across multiple files, like tables (DBF/FPT/CDX) or class libraries (VCX/VCT). The various lists of extensions available include the primary files, but not the supporting files.
Template	XML web services type	Specifies a text merge template used for inserting the necessary code to instantiate and use a web service. If empty, a default text merge template is used.
Class library	XML web services type	Specifies the class and class library used when a web service from this category is dropped on a form or class.

Table 3. *Continued*

Property	Applies to	Notes
Base class	XML web services type	Indicates the base class of the class specified for the Class library item. Read-only when a class and class library are specified. When no class and class library are specified, this item is a drop-down list of base classes, but as far as we can tell, changing it doesn't affect the behavior of items in the category.
Object name	XML web services type	The name used for web services dropped on forms or classes.
Properties	XML web services type	Property settings to use when web services are dropped on forms or classes. (See "Setting instance properties" later in this chapter.)

Categories of the XML web services type have some properties that belong to individual items in other types of categories. That's because the XML web services type of category is dynamic, and there's no opportunity to specify these characteristics for the individual web services.

Working with items

While categories are helpful for organizing the Toolbox, it's really the items within them that make the tool useful. There are several ways to add items to the Toolbox, and quite a few things you can do to customize their behavior once you've added them.

The Toolbox supports five types of items, listed in **Table 4**. Although most have a connection with a particular category type, you can put any type of item into any category.

Table 4. *Item types. The Toolbox allows you to add a variety of item types.*

Item type	Notes
ActiveX control	All registered ActiveX controls are eligible.
Class	Class libraries are added as a whole.
File	Behaves in accordance with Registry settings.
Script	Executes specified code and inserts return value.
Text scrap	Inserts text where dropped. May use text merge.

Adding items to the Toolbox

Class items are added to the Toolbox a class library at a time. You can remove individual classes within a library once you've added the whole library and you can hide some classes from a library (see "Customizing the Toolbox" later), but there's no way to add a single class from a class library to the Toolbox.

The most obvious way to add class items is to choose Add Class Library from the context menu. When you do so, the dialog shown in **Figure 5** appears. The category defaults to the one currently expanded, but you can choose any active category. When you specify the class library to add and choose OK, all the classes in the specified library are added to the specified category.

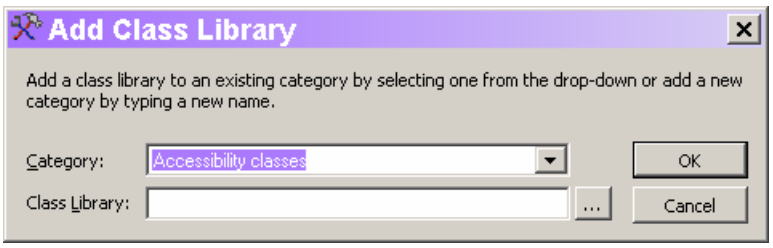

Figure 5. *Adding items to the Toolbox. You add an entire class library at once.*

You can also add classes by dragging the class library from Windows Explorer and dropping it onto the desired category. In this case, the dialog in Figure 5 never appears.

Both techniques for adding class items allow you to add both VCX-based and PRG-based classes.

All types of items can also be added using the Customize Toolbox dialog (shown in **Figure 6**), accessed through the context menu. (The General items in this dialog are discussed later in the section, "Configuring the Toolbox".)

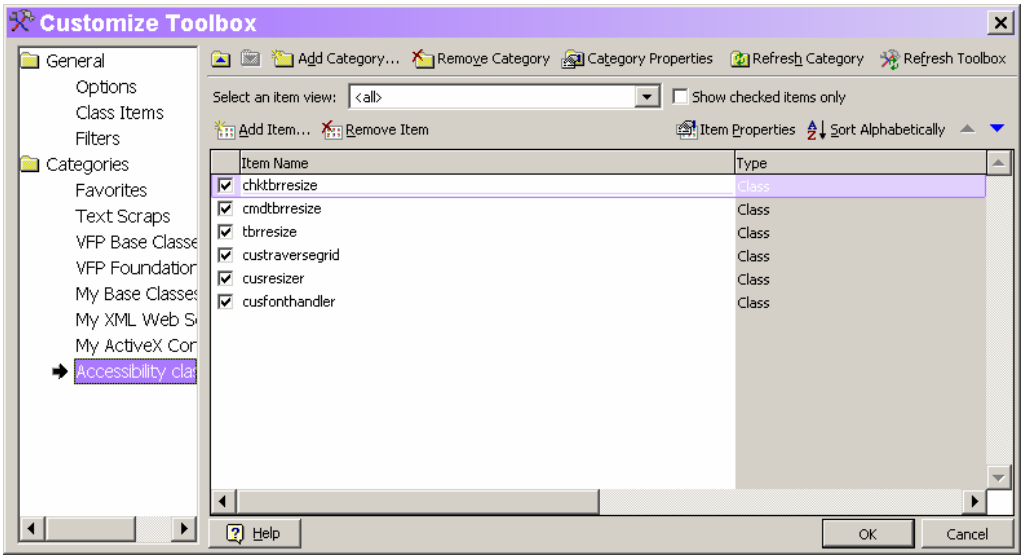

Figure 6. *Adding items to categories. All types of items can be added through the Customize Toolbox dialog.*

To add an item in the dialog, choose the correct category from the left pane, and then click the Add Item button. The Add Item dialog shown in **Figure 7** appears, with the default item type for the category highlighted.

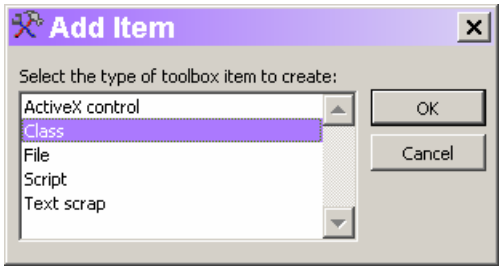

Figure 7. *Adding an item. The first step is to indicate the type of item.*

Once you choose an item type, you specify the item. The exact sequence of steps varies with the Item type. For Class and File, the Open dialog appears to let you specify the class library (VCX or PRG) or file (any type at all).

If you choose ActiveX control, the main body of the right pane changes to display a list of registered ActiveX controls (as shown in **Figure 8**). You can choose one or more to add to the category by checking them. Note also, in this case, the Select an item view drop-down list changes to "ActiveX controls." To return to a display of the categories contents, you need to change it back to "<all>." (For more on this drop-down list, see "Managing categories" later in this chapter.)

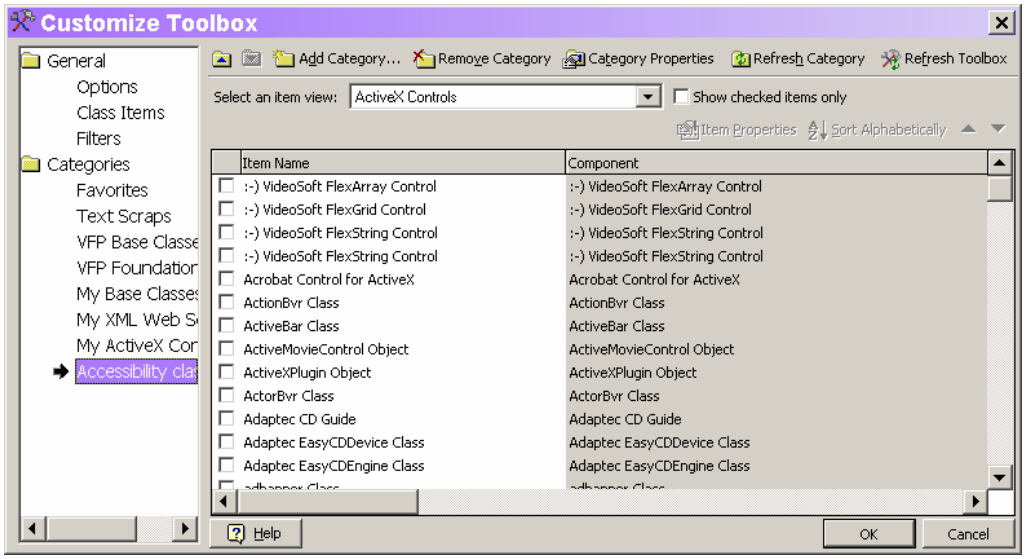

Figure 8. *Adding ActiveX controls. All registered controls are listed. Check the ones you want to add to the category.*

Specifying Script or Text Scrap for the item type opens the Item Properties dialog, which is described in the next section, "Setting item properties."

When a category has the Dynamic folders type, you don't need to add items to it explicitly. All files in the specified folder with the specified extensions are added to the category automatically. If you change the contents of the folder, you need to explicitly refresh

the category. You do so by choosing Refresh Category from the context menu or the Customize Toolbox dialog. (However, dynamic folder categories are all refreshed when the Toolbox is closed and reopened.) Be aware that all items are added to this type of category as File items, even if they contain classes.

Setting item properties

The context menu for items (other than those added automatically to Dynamic folder type categories) includes Properties; choosing it opens the Item Properties dialog. The contents of the dialog vary based on the item type. **Figure 9** shows the dialog for a Class type item, **Figure 10** shows it for a File type item, and **Figure 11** shows it for a text scrap. (The dialogs for the other item types aren't shown here.)

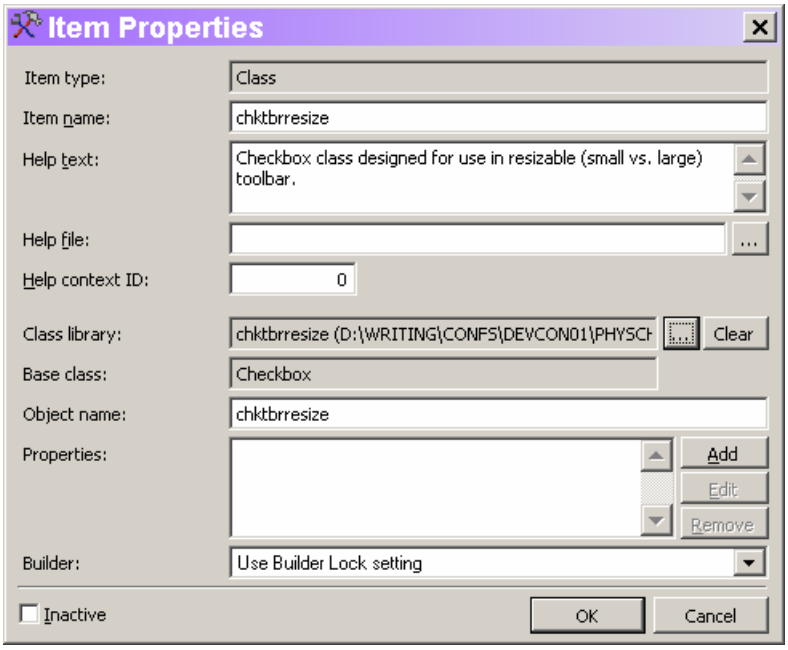

Figure 9. *Specifying item properties. This version of the Item Properties dialog is used for Class type items.*

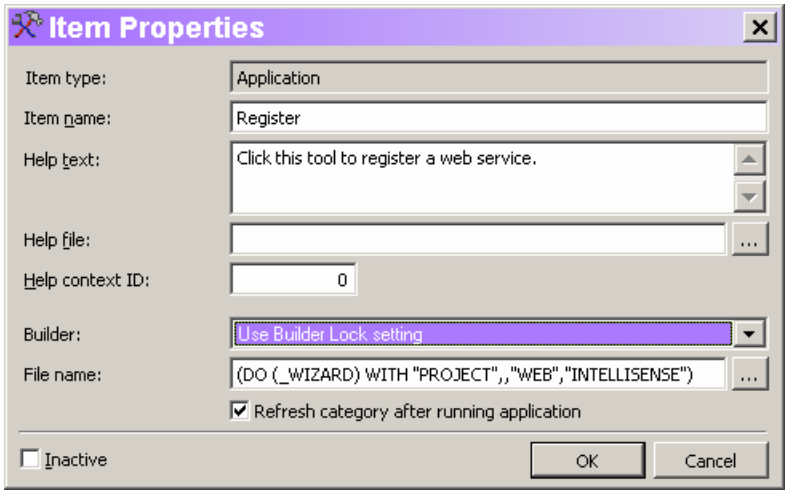

Figure 10. *File item properties. For File type items, the Item type text box shows the type of the underlying file, not just "File."*

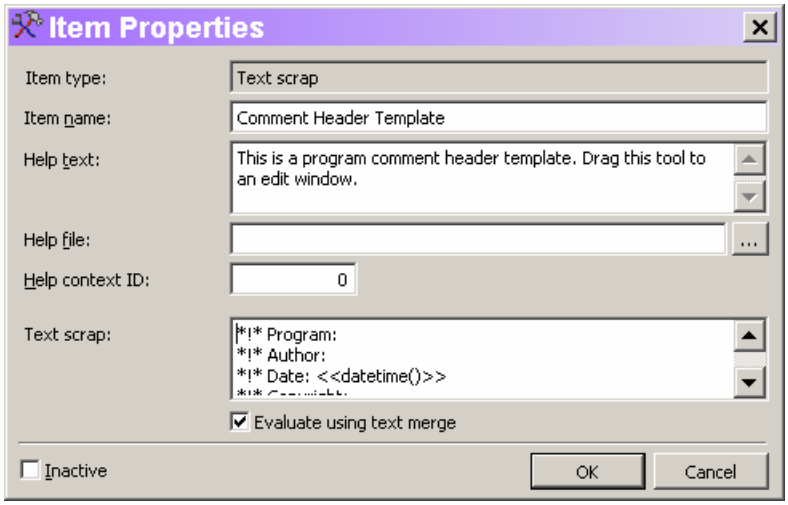

Figure 11. *Specifying text scraps. The properties dialog for text scraps and scripts let you specify the item itself.*

As with categories, many of the properties you can specify are the same for all types, while others apply to only specific types. **Table 5** lists the properties you can specify in the Item Properties dialog.

Table 5. *Item Properties. The list of things you can specify varies with the item's type. Most intriguing is the ability to set property values for instances of classes created from the toolbox.*

Property	Applies to	Notes
Item type	All item types	Read-only; determined by the item type, and for File type items, by the file's type.
Item name	All item types	Specifies the name that appears for the item in the Toolbox.
Help text	All item types	Specifies the text shown in the Help pane at the bottom of the Toolbox when the item has focus.
Help file	All item types	Specifies a Help file containing Help for this item.
Help context ID	All item types	Specifies the help context id for the topic that should be displayed when Help opens. If this is specified and the Help file property is empty, it refers to the VFP Help file.
Inactive	All item types	Indicates whether the item should be displayed in the Toolbox.
Class library	Class, ActiveX control	Specifies the class library that contains the class or the OCX file that contains the ActiveX control. Read-only; for Class items, an ellipsis button allows you to choose the class and class library.
Base Class	Class	Specifies the class of the item. Read-only; an ellipsis button allows you to choose the class and class library.
Class Name	ActiveX control	The ProgID of the ActiveX control. Read-only.
Object name	Class, ActiveX control	Specifies the name of the control to create when this item is added to a form or class or the name of the object variable when the item is added to a code editor. For a Class item, normally the class name. For an ActiveX item, by default, OLEControl.
Properties	Class, ActiveX control	Specifies property settings to be performed on the fly as the item is added to a form or class. See "Setting instance properties" later in this chapter.
Builder	Class, ActiveX control, File	Determines whether the builder for the item, if one is specified, runs when the item is opened. There are three choices: Use Builder Lock setting, Always invoke Builder, or Never invoke Builder.
File name	File	The name of the associated file. This setting can actually be a call to a program. (See the Register item in the My XML Web Services category for an example.)
Text scrap	Text scraps	Specifies the text to be inserted when the item is used. May include items to be evaluated via text merge.
Evaluate using text merge	Text scraps	Indicates whether the text merge should be performed on the text scrap before inserting it.
Script	Script	Specifies the code to be executed when the item is used. The return value of this code is inserted where the item is dropped.
Complete drag script	Script	Specifies code to be run after dropping the item. The return value isn't used, but you can perform interface or clean-up tasks.

Using items

In "Working with the Toolbox" earlier in this chapter, we described the most common things you do with Toolbox items, double-click or drag and drop them to add controls to forms and classes. However, the Toolbox actually supports far more than this.

The context menu for each item varies depending on the item type. For classes other than the VFP base classes, it includes Modify, which opens the Class Designer. For classes that can be subclassed visually, the menu includes Create Subclass, which opens the New Class dialog. For ActiveX controls, the context menu has Open in Object Browser. For form classes, the context menu includes Create Form, which opens the Form Designer with a new form of the specified class. When any forms or container classes are open in the Form Designer or Class Designer, an Add to item is added to the context menu for many items. It points to a submenu showing each available container and form. Choosing one of those items drops an instance of the class on the chosen container or form.

The context menu for Text Scraps includes Copy to Clipboard. For File type items, the choices vary with the file type, but most common are Modify and Run. When you choose Modify, it opens the file in its native editor (even if it's a different application).

File items are displayed using hyperlinks. Clicking on a File item is equivalent to choosing Modify for that item and opens it in its native editor.

Most items have Rename and Delete options on the context menu that do as their name suggests. (Of course, deleting an item from the Toolbox doesn't actually remove it from the class library. Similarly, Rename affects the name you see in the Toolbox, not the actual name of the class.) In addition, the context menu for all items includes Add to Favorites. Choosing it adds the item to the Favorites category without removing it from the category it's already in. This is an easy way to have overall organization, but put the items you use all the time together.

Setting instance properties

The Item Properties dialog for Class items (shown in Figure 9) and for ActiveX items includes a Properties section. This section lets you specify properties to be set in the Property Sheet when the item is added to a form or class. When you click the Add button, the Set Object Property dialog (shown in **Figure 12)** appears. (For ActiveX controls, the left-hand control is a text box rather than a drop-down list.)

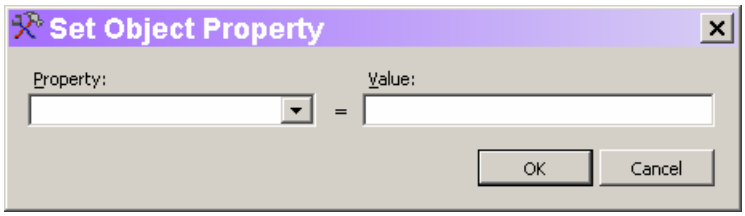

Figure 12. *Setting properties on the fly. The settings you perform in this dialog are applied when you add the item to a form or class.*

To set a property, choose it from the drop-down list or type in the name, and then specify the value you want in the Value text box. Note there's no IntelliSense or other help for adding values here. You need to know what the appropriate values are. After you click OK, the property setting displays in the Properties area of the Item Properties dialog (as shown in **Figure 13**). Once property settings are added there, you can use the Edit and Remove buttons to modify or delete the settings. Double-clicking on a setting in the list opens the Set Object Property dialog with that setting entered.

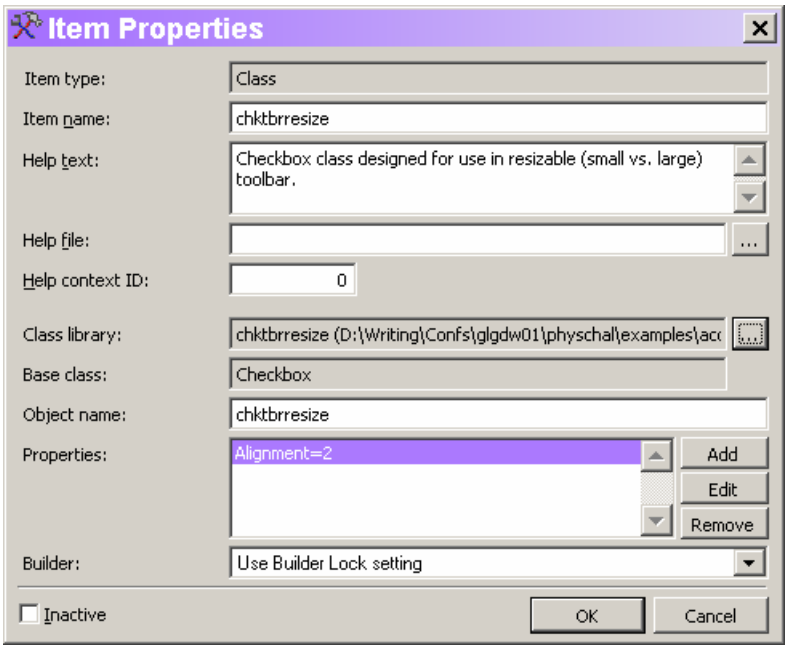

Figure 13. *Instance property settings. After you add property settings, they're displayed in the Properties edit box, where they can be edited or removed.*

It's important to understand that setting properties in this way does not create new subclasses. It's more like a builder in that it sets properties for a particular instance. Don't use this capability when a subclass is appropriate.

However, there are two fairly cool things you can do by setting item properties. First, the value you specify can actually be code. For example, you might set Caption to a value of "(INPUTBOX('Specify caption'))"; when you add the control to a form or class, you'll be prompted to enter the caption. Note the extra parentheses around the function call; they're needed to force the value to be evaluated rather than just assigned to the property. However, don't include the outer quotation marks.

In addition, the property listings for member controls (Column Header, Grid Column, Option Button, and Page) include some special properties that determine what happens when you drop one of these controls onto something other than its normal container. Normally, the appropriate container is added based on the VFP base classes, and then an object of the specified class is added to it. The ContainerClass, ContainerClassLib, and MemberCount properties let you determine what class the container is based on, and how many objects of the chosen class are added. These properties do not appear anywhere except in this dialog.

Customizing the Toolbox

Given everything we've already described, you may be wondering what else we can possibly do to customize the Toolbox. In fact, there are some more global changes you can make.

First, the context menu contains several toggles that affect the Toolbox as a whole. Builder Lock determines whether builders appear for items set to follow the Builder Lock

setting. Items in the Toolbox ignore the global Builder Lock setting from the Options dialog on the Tools menu. The setting for the Toolbox is independent of the Options dialog setting.

Display Help Text determines whether the Help pane appears at the bottom of the Toolbox. Always on Top determines whether the toolbox stays on top or drops behind other forms.

Most configuration of the Toolbox, though, is performed in the Customize Toolbox dialog (shown in Figure 6 and Figure 8). In "Adding items to the Toolbox" earlier, we discussed one use for this dialog, but it's far more capable than just editing the list of items in a category. The top section of the left pane provides access to items that affect the Toolbox as a whole.

Configuring the Toolbox

Figure 14 shows the General Options view of the dialog. This section contains a variety of settings for Toolbox behavior, including the three that can be set from the context menu. Run associated builders is the same as the Builder Lock context menu item. Display Toolbox help text section corresponds to the Display Help Text item, and Always on top is a mirror of the Always on Top context menu item.

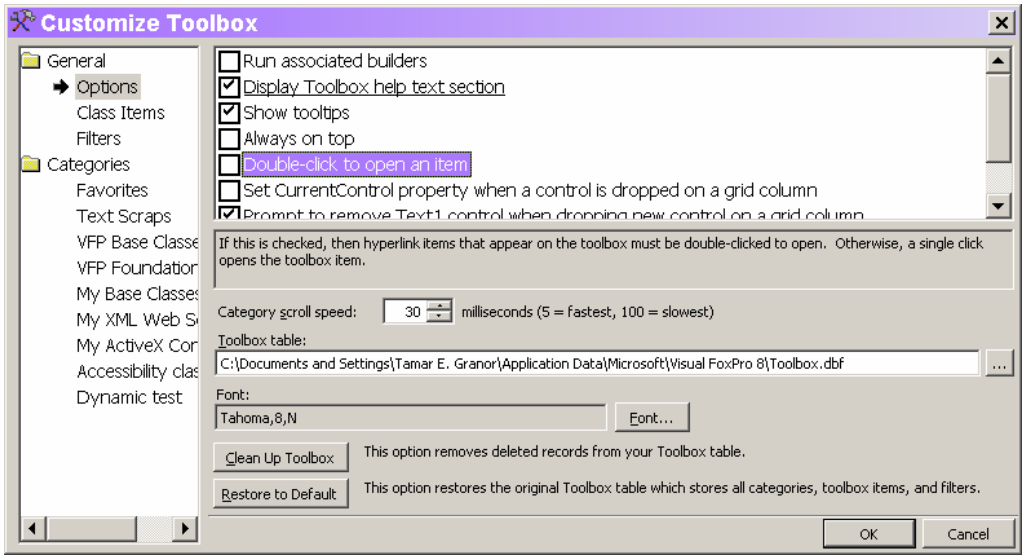

Figure 14. *Setting Toolbox behavior. The Options view of the Customize Toolbox dialog contains settings that affect the Toolbox as a whole.*

Show tooltips determines whether tooltips are displayed when the mouse hovers over an item. The tooltips use the Help text specified for the items. If space is at a premium for you, turn off the Help text section and leave tooltips on.

Double-click to open an item applies to File items. By default, a single click on a File item opens it in its native editor. When this item is checked, it takes a double-click. If you find you often open items by accident, consider changing this setting.

The next two items, Set CurrentControl property when a control is dropped on a grid column and Prompt to remove Text1 control when dropping a new control on a grid column,

combine to determine what happens when you drop controls on grids. When the Prompt item is checked, dropping a control on a grid column that contains a default text box named Text1 brings up the dialog shown in **Figure 15**. If you answer Yes, not only is the control you're dropping added to the column, but the text box is deleted. When the Set CurrentControl option is checked, the newly added control becomes the current control for the column. When this option is not checked, CurrentControl isn't set to the new control, even if you delete the default text box.

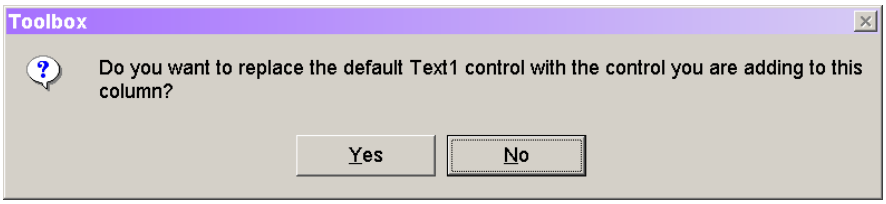

Figure 15. *Replacing grid controls. When you drop a control onto a grid column, this dialog lets you get rid of the default text box.*

The final item in this section, Allow Toolbox to be minimized, adds a minimize button to the Toolbox and enables the control box, putting Minimize on that list as well.

As noted in the section "Working with the Toolbox" earlier in this chapter, holding the mouse over the arrows at the top and bottom of a category scrolls within that category. The Category scroll speed spinner lets you determine how fast the items scroll.

As you'd expect, the Toolbox table item points to the table that contains the Toolbox data. Among other things, the ability to change this setting means you can maintain several toolbox data sets (perhaps for different projects) and switch among them. (See "Filtering the Toolbox" later in this chapter for an alternative approach.)

If the 8-point Tahoma the Toolbox uses by default doesn't work for you, use the Font button to choose your favorite font and size. (Unfortunately, it doesn't change the font in any of the associated dialogs.)

Finally, the two buttons at the bottom of this page let you clean up after yourself. Clean Up Toolbox packs the Toolbox data to get rid of deleted records. Restore to Default restores the original toolbox, though it gives you the option of keeping categories and items you have added. In that case, all it does is restore the original items to their original settings (for example, removing property settings you've added). A copy of the Toolbox table is saved as a backup.

Determining the behavior of class items

The Class Items view of the dialog (**Figure 16**) lets you determine how Class items are named when added to the Toolbox and what happens when you drag Class items to code editing windows.

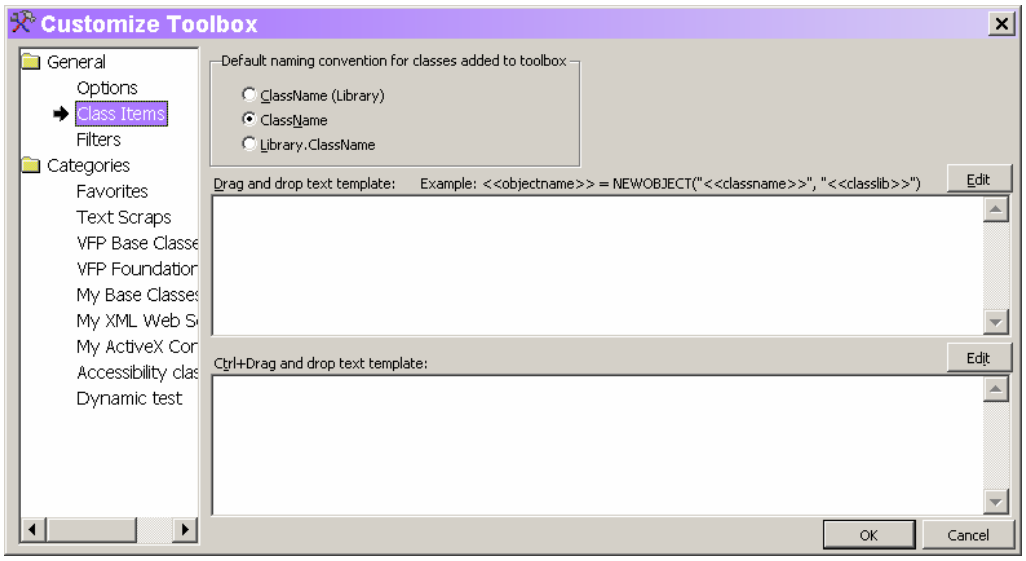

Figure 16. *Class item behavior. This page of the Customize Toolbox dialog determines default naming of Class items and their behavior when dropped into code windows.*

When you add a class library to the Toolbox, each item is given a name. The Default naming convention option buttons determine the format of that name. (Keep in mind you can change the name in the Item Properties dialog or on the appropriate category's page in the Customize Toolbox dialog.) For a class called cmdMyButton in a class library named MyClasses, the three options are:

- cmdMyButton (MyClasses)
- cmdMyButton
- MyClasses.cmdMyButton

The two edit boxes on this page control what happens when you drag a Class item onto a code editing window. The Drag and drop text template is used for a normal drag, while the Ctrl+Drag and drop text template is used for a drag with the Ctrl button held down. Whatever text you put into the edit box is evaluated using text merge, and then dropped into the code editing window.

By default, dragging a VFP base class generates code like:

```
Text = CREATEOBJECT("Textbox")
```

Dragging any other Class item results in code using NewObject(), like this:

```
_checkbox = NEWOBJECT("_checkbox", "_BASE.VCX")
```

In both cases, the name used for the new object is the object name specified in the Item Properties dialog. If this code isn't what you want, write your own template in the dialog. **Table 6** shows the data related to the item you can use in your template.

Table 6. *Creating text templates. These pieces of data about the item you're dropping are available for text merge in templates.*

Name	Contains
BaseClass	The base class of the item.
ClassLib	The class library of the item, with complete path.
ClassName	The name of the item's class.
ObjectName	The name specified for the object in the Item Properties dialog.

For example, to create documentation for a class, you might put this template in the Ctrl-Drop template:

```
Base Class: <<BaseClass>>
Class Library: <<ClassLib>>
Class Name: <<ClassName>>
Object Name: <<ObjectName>>
```

Filtering the Toolbox

The third page of the General section of the Toolbox (shown in **Figure 17**) lets you define and manage filters that limit the categories displayed in the Toolbox at any time. When any filters are defined, the context menu adds a Filters item that opens a submenu listing all defined filters. Choose a filter from that list and the Toolbox displays only the categories specified for the filter. We can see using filters to manage the different class libraries needed for different clients or different projects.

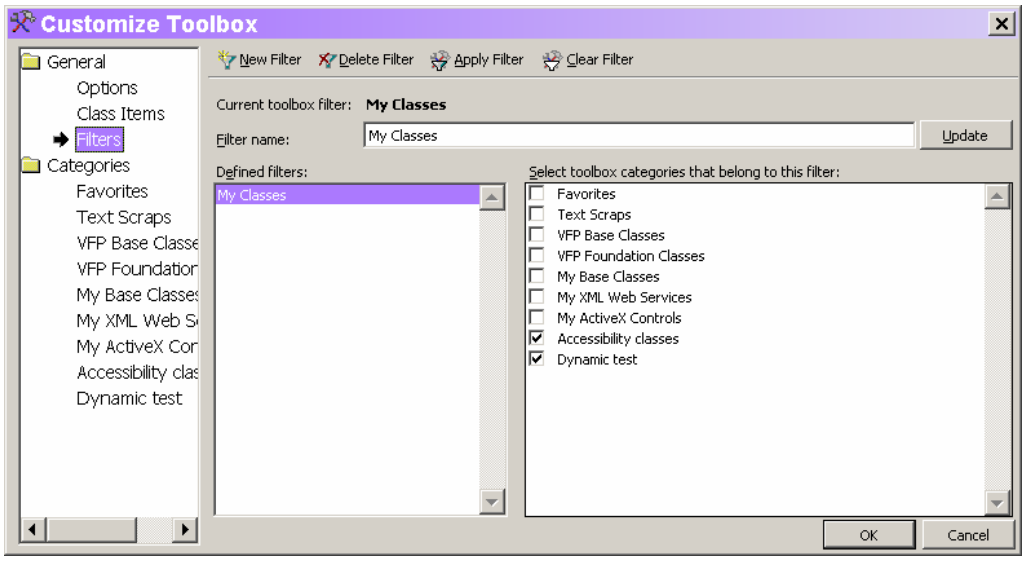

Figure 17. *Managing filters. A filter contains one or more categories.*

To define a filter, click the New Filter button. The new filter is added in the dialog and focus is set to the Filter name textbox. Type the name for your filter, and then check the categories that should be displayed for the filter. The Update button updates the name of the filter in the Defined filters list.

You can edit any filter by clicking on it in the Defined filters list and changing the chosen categories. As you'd expect, Delete Filter removes a filter from the list.

Apply Filter and Clear Filter let you set and remove a filter, but you can do that right from the context menu as well.

Managing categories

The Categories pages in the Customize Toolbox dialog let you manage the list of categories and their contents. Techniques for adding items to categories are discussed in "Adding items to the Toolbox" earlier in this chapter.

When Categories or any specific category is chosen, a button bar at the top of the dialog (shown in **Figure 18**; see Figure 6 and Figure 8 for the whole dialog) lets you rearrange the list of categories, add, remove and edit categories, and refresh an individual category or the whole Toolbox.

Figure 18. *Category management. This button bar appears in the Customize Toolbox dialog when a category is chosen.*

The Select an item view drop-down list controls what appears in the main pane of the dialog. The choices in the drop-down list are "<all>," "Visual FoxPro Class Libraries" and "ActiveX Controls." The button bar beneath the drop-down list changes based on the choice in

the drop-down list. When "<all>" is chosen, it contains buttons (**Figure 19**) for adding, removing, and editing individual items, as well as sorting and rearranging items. When "Visual FoxPro Class Libraries" is chosen in the drop-down list, the button bar (**Figure 20**) contains items for managing entire class libraries. When "ActiveX controls" is chosen, no buttons are available.

Figure 19. *Managing items in the Customize Toolbox dialog. When the item view is set to "<all>," the button bar lets you work with individual items. The up and down arrows let you rearrange items in the category.*

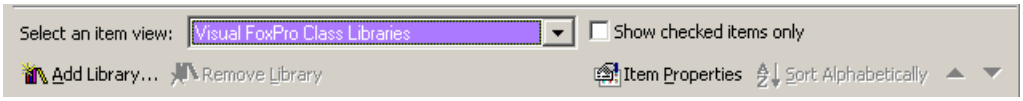

Figure 20. *Working with libraries. When the item view is set to "Visual FoxPro Class Libraries," the buttons manage entire libraries.*

In any view, check or uncheck items to show or hide them in the Toolbox. The Show checked items only check box in the dialog determines whether you see the hidden items while in the Customize Toolbox dialog. Also, no matter what view you're in, the Item Name column is editable—changing it determines what appears for the item in the Toolbox.

Working with the source code

As with other tools written in VFP, the source code for the Toolbox comes with the product. Source for all of the "Xbase tools" is in a file called XSource.ZIP in the HOME() + "Tools\XSource" directory. When you unzip it (make sure to select the Use folder names option), a new directory called VFPSource is created. Each tool has a subdirectory within that directory. The Toolbox project is in a Toolbox directory.

You can modify or subclass any of the code used for the Toolbox and rebuild the application. By default, the Toolbox is Toolbox.APP in the HOME() directory. You can change that by setting the _TOOLBOX system variable.

By default, Toolbox data is stored in a table called Toolbox.DBF in the application data directory (the one indicated by HOME(7)). As noted in "Configuring the Toolbox" earlier in this chapter, you can change the location and file name.

The Toolbox table contains one record for each category and one for each item in the Toolbox. **Table 7** shows the fields in the Toolbox table.

Table 7. *Toolbox data. The Toolbox table contains one record for each category and one for each item in the Toolbox.*

Field	Type	Purpose
UniqueID	Character	A unique identifier for the record. Recommended format is "vendor.ID". For records added through the Toolbox interface, "user" is filled in for the vendor portion and SYS(2015) is used to provide the ID portion.
ShowType	Character	Indicates the type of item. For example, "C" for category and "T" for (tool) item. See Toolbox.H in the Toolbox project for the complete list.
ToolTypeID	Character	The type of item. For categories, in the form "Category.CategoryType". For items, just "ItemType".
ToolType	Memo	The type of item in readable format. This is the type that appears in the Properties dialog.
ParentID	Character	For items, the UniqueID of the containing category.
ToolName	Character	The descriptive name of the category or item, as it appears in the Toolbox.
ImageFile	Memo	The icon to use for the item. Not used for categories.
ClassType	Character	Indicates the type of item. "CATEGORY" for categories, "CLASS" for class items, and so forth. For file items, the file type, such as "APP."
SetID	Memo	For class items, the containing class library. (Important because class libraries are added and, sometimes, removed as a whole.)
ClassName	Memo	The name of the class used to process this item. For example, by default, "_classtool" for class items and "_activetool" for ActiveX items.
ClassLib	Memo	The class library containing the class named in ClassName. If empty, _toolbox.vcx is used.
ToolTip	Memo	The help text for the category or item. This appears in the Display Help pane and as a tooltip.
HelpFile	Memo	The Help file (including path) containing Help for this item.
HelpID	Numeric	The help context ID for this item in the Help file specified by HelpFile (or in the VFP Help file, if HelpFile is empty).
ToolData	Memo	Data for this category or item. For items, includes the class name, class library, base class, and object name information, as well as the list of properties to set in this instance. For file items, includes the file name. For text scraps, includes the text to insert. For scripts, includes the script. For ActiveX controls, includes the OCX file name, the class name and the object name.
DisplayOrd	Numeric	The display order of this item. For categories, the position in which the category appears. For items, the position of the item in the category.
LockAdd	Logical	For categories, indicates whether items can be added. (True indicates no additions are permitted.) Ignored for items.
LockDelete	Logical	Indicates whether the item or category can be deleted. When this is set to True, Delete does not appear on the context menu and the Remove Item button is disabled in the Customize Toolbox dialog.
LockRename	Logical	Indicates whether the item or category can be renamed. When this is set to True, the Rename item does not appear on the context menu.
Inactive	Logical	Indicates whether the item or category is currently displayed in the Toolbox. (True means the item or category is not displayed.)
User	Memo	Available for user-specified information.
Modified	DateTime	Time stamp of last change to item or category.

Finally, when the Toolbox is running, the variable _oToolbox contains an object reference to it. The oToolboxEngine property points to the object responsible for much of the Toolbox's behavior.

Summary

The Toolbox is an amazingly capable new tool. It offers the power of the Component Gallery without its complexity. We plan to use the Toolbox exclusively and never open the Form Controls toolbar again.

Incredibly, the Toolbox is just one of the new tools in VFP 8. The next two chapters will explore the other two, Code References and the Task Pane Manager. Chapter 5, "Better Tools," looks at changes to existing tools.

Updates and corrections to this chapter can be found on Hentzenwerke's web site, **www.hentzenwerke.com**. Click on "Catalog" and navigate to the page for this book.