Contents

Introduction

VB AppFramework (abbreviated to AFW throughout much of this document) is a unique and powerful add-in to the Visual Basic 4   (16 or 32 bit edition) Windows development environment that simplifies and organizes the way you work.   AFW lets you create templates from Visual Basic objects, and then allows you add these templates to a project at any time.   AFW also lets you create styles from groups of properties, and apply these styles at any time.   Finally, AFW lets you create entire application frameworks by grouping together related templates.

AFW is extremely flexible.   Whether you work alone on your Visual Basic projects, or as part of a large corporate team, youll find collecting and using Visual Basic templates and styles is a real time-saver.

These collections of objects are saved away by AFW into libraries.   Each library is actually a single standard Microsoft Access database file.   You can create your own library, purchase commercial libraries, or share a library with co-workers.

AFW leaves you completely in control.   No unexpected or automatic features are added to the Visual Basic environment; you decide when and where to use the enhanced features AFW provides.   When youre ready, built-in wizards walk you through the process of capturing your objects, or through the steps of adding or applying them to your Visual Basic projects.

AFW was designed to be tightly integrated with the Visual Basic environment.   The terms, forms, objects, and even toolbar buttons have been designed to make you feel right at home.

AFW was created *by* VB professionals *for* VB professionals.

This topic shows you how to set up VB AppFramework on your computer, and attach it to your Visual Basic environment.

Contents

16 and 32 Bit Versions

This document discusses the functionality of VB AppFramework. This product is shipped with both a 16 and 32 bit version. While the versions are functionally equivalent, there are some interface and usability differences between them.

The most significant usability difference is the fact that the 16 bit version uses the *Outline* control to display information in the wizards and browser, whereas the 32 bit version uses the *TreeView* control for the same purpose. Throughout this document, the supporting graphics show the TreeView control. The organization of information is the same in the Outline, but with less granularity as to the type of template, or style that is being displayed.

The second significant difference is that the 32 bit version is implemented as an *In-Process* OLE server, whereas the 16 bit version is an out-of-process OLE server. In-Process servers have significant performance improvements over out-of-process servers. In addition, all forms within In-Process servers are modal to VB, whereas out-of-process forms are not.

Assumptions

It is assumed throughout this document and while using AFW that you have a working knowledge of Microsoft Windows and Visual Basic.   Oftentimes VB terms are mentioned in this help file, but not thoroughly explained.   Please refer to your VB documentation for clarification of these terms.

Setting Up

VB AppFramework is an Add-In to the Visual Basic Integrated Development Environment (VBIDE), and is not a stand alone Windows application. Nevertheless, the initial setup of AFW proceeds in the same way as with most stand alone applications.

1. Insert the AFW disk titled Setup in your floppy disk drive.
2. Click the *Start* button on the task bar, then click the *Run* icon (Windows 95), or select *Run* from the *File* menu in File Manager (earlier versions of Windows).
3. Type A:\SETUP16\SETUP to start the 16 bit setup program or A:\SETUP32\SETUP to start the 32 bit setup program.
4. Follow the instructions as they appear during the setup process.

The initial setup process will copy the AFW files into the directory C:\AFW for the 16 bit setup or C:\VB AppFramework for the 32 bit setup, assuming you opted for the suggested drive and directory during setup. It also will create two directories underneath this directory: \LIB, and \REPORT. The \LIB directory is the default directory where your libraries are kept. The \REPORT directory contains Crystal Reports templates. An ini file, AFW.INI, is created in your Windows directory, or the information is placed in your system registry depending on the version of Windows you are using. Finally, entries are automatically created in the [Add-Ins 16] and [Add-Ins 32] sections of your VB.INI file. These entries are required for all Visual Basic aware Add-In applications.

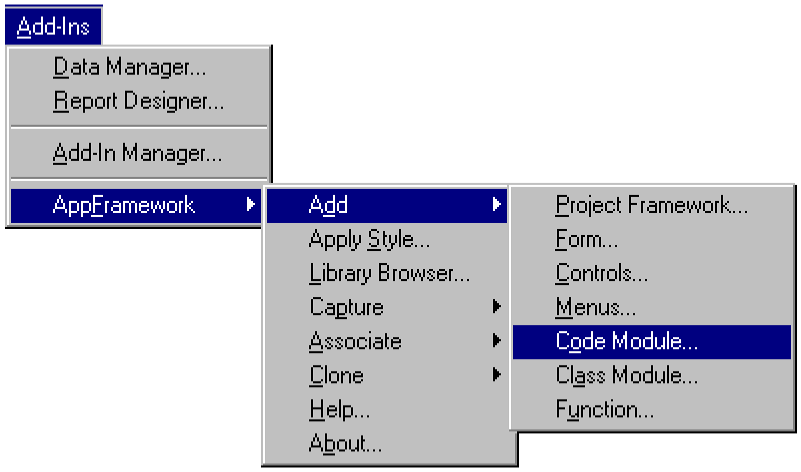
Attaching to VB

To attach AFW to the Visual Basic IDE, you must use the Add-Ins menu to locate and mark it as a currently active Add-In. Once activated, AFWs menu items will appear in Visual Basics Add-Ins menu, and all features will be available for your use. Activating AFW is easy to do:

1. Select *Add-In Manager* from Visual Basics *Add-Ins* menu.
2. Locate *VB AppFramework.Application* in the *Available Add-Ins* list and click in the little box to mark it for activation..
3. Click the *OK* button to complete the activation.

You can verify AFW is active by looking at Visual Basics Add-Ins menu. A main menu and several new AFW sub-menu items will appear in this menu. Each of these menu items will be described in detail later on. The figure shows Visual Basics Add-Ins menu after VB AppFramework has been attached.

Note Visual Basic Add-Ins remain active until you de-select them using the Add-Ins Manager, even if you load different Visual Basic projects or reboot the computer. VB AppFramework will always be just a click away!



VB AppFramework Menus

Features of VB AppFramework

VB AppFramework provides a mechanism for *capturing* Visual Basic Templates and Styles, a repository called a *Library* for storing them, and a mechanism for *adding* or *applying* them to your Visual Basic projects. The Library is a Microsoft Access database file, structured as an AFW Library. Since each Library is a self-contained MDB file, it can easily be replicated and distributed. AFW lets you create and populate your own Libraries and also ships with Libraries that are ready to use. Wizards help you capture, find, and use Templates and Styles contained within these Libraries.

Tip To keep you from confusing your AFW Libraries with other Microsoft Access databases and applications you create, AFW uses an the file extension .AFW instead of the .MDB extension normally assigned by Access. Dont worry, though, we havent modified the Access database format in any way.

The following table summarizes the features of VB AppFramework

Feature	Description
Library Browser	Central tool that allows you to browse AFW libraries; create and install libraries; report on contents of libraries; edit template names and comments.
Capture Wizards	Capture templates and styles into an AFW library; associate custom controls, references, resources, and files and create complete project frameworks.
Frame Wizard	Add an entire application framework; collections of forms, code modules, class modules, custom controls, references, and resources; to a VB project; place associated files in the project directory.
Form Wizard	Add a form template to a VB project.
Control Wizard	Add one or more control templates to a VB form.
Menu Wizard	Add one or more menu templates to a VB form.
Class Wizard	Add a class template to a VB project.
Code Wizard	Add a code template to a VB project.
Function Wizard	Add a complete function, or any fragment of code to a VB code window.
Style Wizard	Apply styles to forms, or controls.
Associate Custom Controls	Associate custom controls to a project for the purposes of creating a project framework.
Associate References	Associate references to a project for the purposes of creating a project framework.
Associate Resource	Associate a resource file to a project for the purposes of creating a project framework.
Associate File	Associate any file to a project for the purposes of creating a project framework.
Clone Controls and Forms	Instantly copy one or more controls *with their methods*, or create an exact copy of a form.
Clone Styles	Instantly create a copy of a style and apply it to similar objects.

Product Support Services

Semiotix Systems has worked closely with Microsoft to bring you the very best product possible. Your comments and questions about VB AppFramework are welcome.

You may contact us by telephone at:

Product Support:	(970) 339-7157
Telephone:	(303) 743-1400
Fax:	(303) 743-1410

Or you can write to us at:

Semiotix Systems, LLC
10620 East Bethany Drive
Aurora, CO 80014

Or you can send us e-mail at:

afw@semiotix.com
76450,3577@compuserve.com

Your First AFW Library

In this topic we will provide a step-by-step set of instructions for creating and using a new Library. We will first create a new Project Framework in this Library and capture a Form Template, A Controls Collection, and Menus Collection into the new Framework. We will then use our Templates by adding the Form to a Visual Basic project, then adding the Controls Collection and Menus Collection Template to a form. Finally, well capture a Style into the Library and apply it to the controls on a form.

You might think of this topic as the equivalent of creating a simple Hello World! application in Visual Basic. You wont learn everything in this topic, but you will quickly form a solid foundation of your AFW understanding.

Contents

Starting AFW

Remember that AFW is an Add-In to the Visual Basic IDE.   You start AFW by simply starting up Visual Basic.   You will need to setup the product and attach it, as described in the previous topic, but you need only do this once.   After this initial setup and attachment, AFW is always just a click away when youre working in Visual Basic!

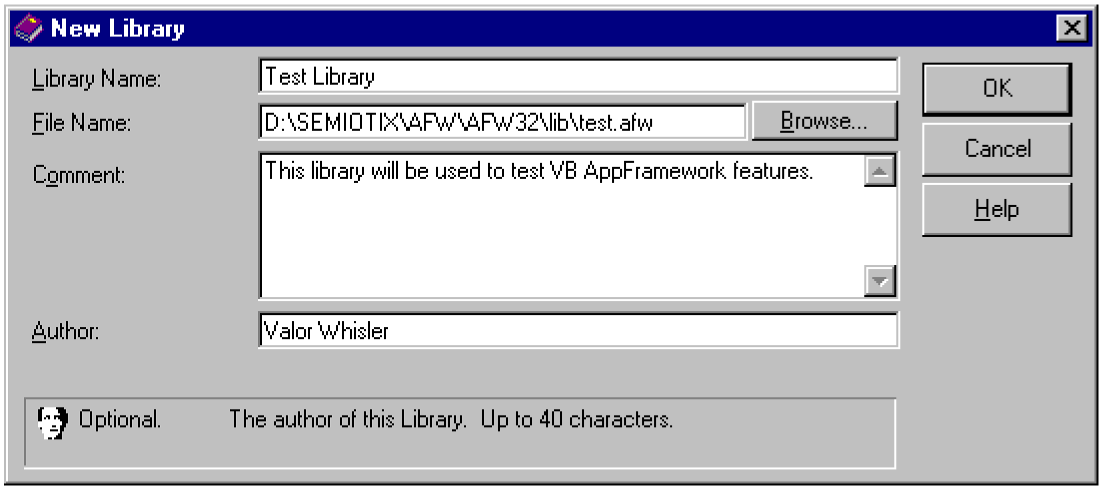
For this exercise, youll need to first create a new Visual Basic project and save it (use any project name you choose).   The project should have the default form titled Form1. Leave the project open.

Creating a New Library

To create a new AFW library select *AppFramework* + *Library Browser* from the Visual Basic *Add-Ins* menu. The Library Browser is where you create, install, remove, delete and maintain AFW Libraries.

Step 1

Select *Library* + *New* from the menu. The New Library dialog shown in figure below will be displayed. This dialog lets you enter the file name for the new Library, comments to describe the Library and the name of the author. The name, comment, author and a date and time stamp are stored in the Library, providing useful information when you work with more than one Library.



Library Name: Test Library
File Name: D:\SEMIOTIX\AFW\AFW32\lib\test.afw
Comment: This library will be used to test VB AppFramework features.
Author: Valor Whisler

Optional. The author of this Library. Up to 40 characters.

Describing the New Library

For this exercise, enter the following:

- Library Name - Test Library
- File Name - yourpath\NewLib.afw, where *yourpath* is the directory you choose to store your AFW Libraries.
- Comment - This Library will be used to test VB AppFramework features.
- Author - your own name.

This information will show up in the Library Browser to make your Library easy to identify. Click *OK* to complete the creation of the new Library.

The new Library is created from a database template that was placed in your \LIB directory during the setup process. Each new Library is seeded with a description of all standard Visual Basic controls, a set of Default Styles (see the discussion later in this topic under *Capturing a Style*), and a Project Framework entitled Public Project Framework.

Making the Library Active

Step 2

To use a Library in the AFW Wizards, you must first make it the *Active Library*. To do this, highlight the Library you have just created and select *Library + Activate*. This Library and all of its Templates, Styles and Project Frameworks is now available for use.

Creating a New Project

Step 3

Next, well create a new Project Framework called   Test Project in our Library.   In the Library Browser, highlight the Library you just created in Step 1.   Now select *Project + New* from the Browser menu.   The *New Project Framework* form appears.   Fill in the fields to identify the new Project Framework as follows:

- Project Name -   Test Project.
- Comment - any comment you want.
- Author - your name.

Click *OK* to create the Project Framework.

Capturing a Form Template

Step 4

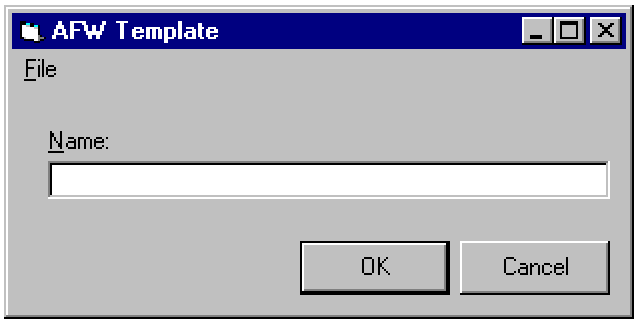
At this point, your new Library and Project Framework contain no Templates or Styles. Before we can capture Templates or Styles, however, we need to create some objects in your active Visual Basic project to work with.

In Form1 of your active Visual Basic project, do the following:

- Add two command buttons, labeled OK and Cancel.
- Add a text box.
- Add a label for your text box called Name.
- Set the Form1 Caption to AFW Template.
- Create a top-level File menu for the form along with the following sub-menus: New, Open, Save, Print, and Exit. Add some simple code behind these menus, like a MessageBox command.
- Go into the code window for Form1 and add some comments to the top of the window, such as a copyright notice and a description of the form.
- Save Form1 within Visual Basic.

We could make this form more elaborate, but the above will suffice for this exercise. Your form should look like the following figure.

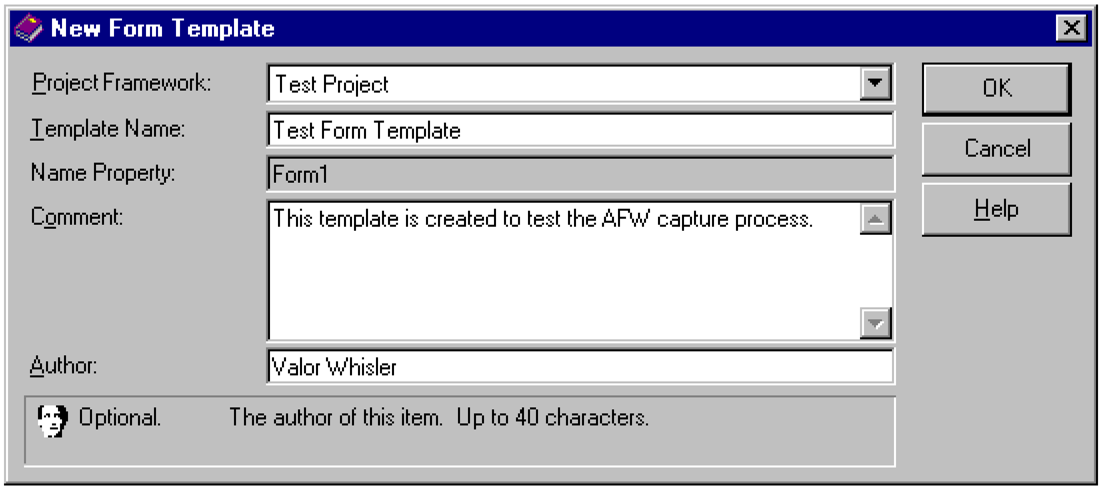
Tip Keep form templates simple, and generic. In this way they will be easy to plug into any project. You may want to create templates for various types of tasks. You may like to design MDI children in one way, dialogs in another way, and data entry forms a third way. Once you have a generic design for such a task, capture it and make it as a Template so its easy to reuse.



The Sample Form Template

Make sure that Form1 is open and has the input focus. Now start the Capture Wizard by selecting *AppFramework* + *Capture* + *Form* from the Visual Basic *Add-ins* menu. A dialog titled *Capture Form Template - Step 1* will be displayed. At this point, there should be just one Project Framework (Test Project) and no Templates contained in the TreeView control on this dialog. The Active Library (Library: Test Library) is noted in the caption above the TreeView. Click *New* to create a new Form Template. This will bring up a dialog titled *New Form Template*.

Select Test Project as the Project Name, enter Test Form Template as the Template Name, a Comment of your choice, and your name as the Author. Note that you cannot edit the Name Property field. This is the name that you assigned to the form in Visual Basic. Your completed dialog should look like the figure below.



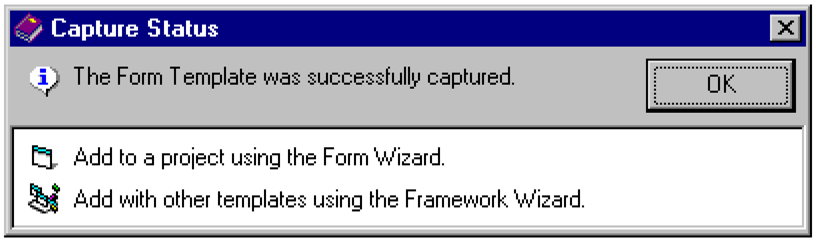
New Form Template

Project Framework:	Test Project
Template Name:	Test Form Template
Name Property:	Form1
Comment:	This template is created to test the AFW capture process.
Author:	Valor Whisler

Optional. The author of this item. Up to 40 characters.

Describing the Form Template

Click *OK* to accept your edits. A dialog titled *Capture Form Template - Step 2* appears to provide summary information about the Template you are capturing. Click *Finish* to perform the capture. Once completed, a *Capture Status* dialog like the figure below is displayed. This dialog confirms the successful capture and indicates that your new Template can be used by running the Form Wizard or the Framework Wizard. Click *OK* to return to your work in Visual Basic.



Capture Status

The Form Template was successfully captured.

Add to a project using the Form Wizard.

Add with other templates using the Framework Wizard.

A Successful Capture

Capturing a Collection of Controls

Step 5

Next we will capture a collection of controls as a Template.   A collection is a group of one or more controls.   Our collection for this exercise will be simple; the two command buttons: *OK* and *Cancel*. These buttons and their associated methods are used together on many forms and thus make a logical candidate for reuse.

On Form1, Select the *OK* and *Cancel* buttons and start the Capture Wizard by selecting *AppFramework + Capture + Controls* from the Visual Basic *Add-ins* menu.   The *Capture Control Templates - Step 1* dialog appears. As in Step 4, click *New* and complete the information in the *New Controls Collection* dialog. You will note that the Name Property field in this dialog contains the value <all control names>.   This is because AFW doesnt yet know how many controls you are capturing.   Make sure you have Test Project selected and enter the name OK, Cancel Collection for the name of the controls collection.   Click *OK* to accept the edits and move to the *Capture Control Templates - Step 2* dialog.   Click *Finish* to execute the capture.   A *Capture Status* dialog will be displayed to confirm the successful capture.

Tip   Keep the methods (i.e., the event procedures) for your Control Templates as generic as possible, or make calls to external procedures within your methods.   In this way, Control Templates can be reused widely with little or no modifications.

Capturing a Collection of Menus

Step 6

Lets complete our capture Templates exercises by capturing a collection of menus. Normally, creating menus is a tedious process. AFW can save you lots of valuable time by letting you reuse menus that have been captured into your Libraries.

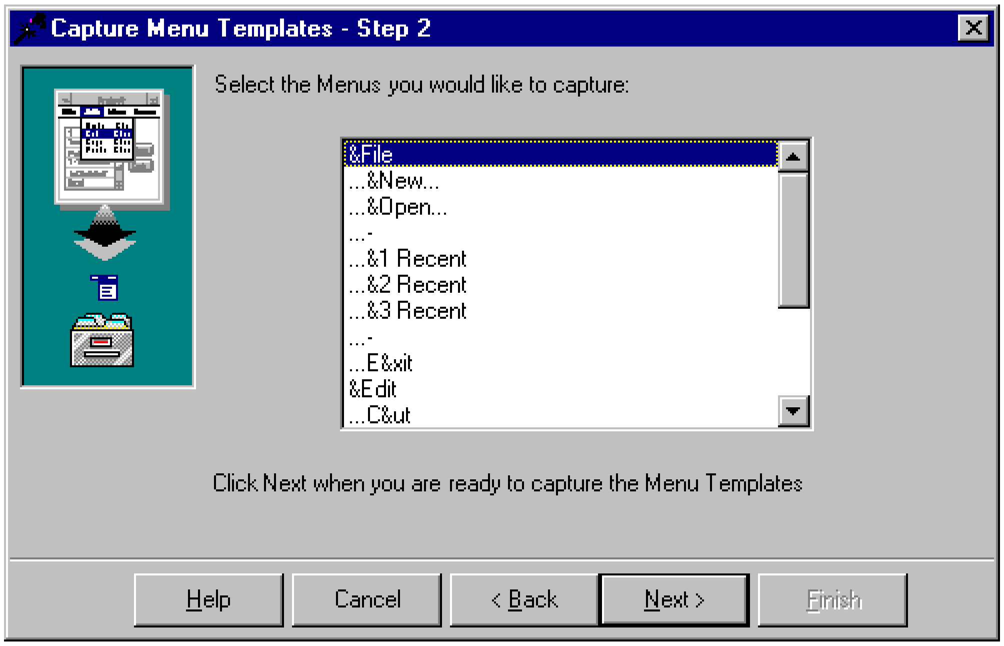
Unlike forms and controls, Visual Basic doesnt provide a way for you to select menus on a form in design time. AFW makes up for this by letting you select menus in the Capture Wizard.

Make sure that Form1 is open and has the input focus. Select *AppFramework* + *Capture* + *Menus* from the Visual Basic *Add-ins* menu. The *Capture Menu Template - Step 1* dialog appears. Click *New*. The *New Menu Templates* dialog appears. Name the collection: File Menus and fill in the other information on the dialog. When you are done with your edits, click *OK* to proceed to the *Capture Menu Templates-Step 2* dialog.

In the *Step 2* dialog, you must select the menus you want to include in the current collection. You can select any number of menus, subject to the following conditions:

1. All menus to be selected must have a root menu. That is, you cant select a sub-menu without also selecting its parent.
2. All menu selections must be contiguous.

Select the entire menu structure. Your dialog should look like the following figure.



Capture Menu Templates - Step 2

Select the Menus you would like to capture:

&File
...&New...
...&Open...
...-
...&1 Recent
...&2 Recent
...&3 Recent
...-
...E&xit
&Edit
...C&ut

Click Next when you are ready to capture the Menu Templates

Selecting Menus to Capture

Click *Next* to proceed to the *Capture Menu Templates - Step 3* dialog, then *Finish* to execute the capture. A *Capture Status* dialog confirming the successful capture will be displayed.

Adding the Form to a Project

Step 7

For this next exercise, lets create a new project within Visual Basic and save it. You can give it any name you want. Leave the project open. Now lets use the Templates we created in Steps 4, 5 and 6 above. Well start by adding a Form Template to your project.

Select *AppFramework* + *Add* + *Form* from the Visual Basic *Add-Ins* menu to start the Form Wizard. This Wizard walks you through the steps required to add a Form Template from a Library into a Visual Basic project.

If you followed the steps above, a Project Framework called Test Project will be displayed in the TreeView on the *Add Form - Step 1* dialog. (If you had other Project Frameworks defined in the Active Library, they would also be displayed here.) Expand the Project Framework node by clicking on the + symbol. You can now see all Form Templates contained within the Framework; in this case, the one Template called Test Form Template. Select this Template. Note that the type of item selected, its author name and last data modified are displayed in the status bar under the TreeView.

Click *Next* to proceed to the *Add Form - Step 2* dialog. At this point, you can change the name of the Visual Basic form and its corresponding file before adding it to your project window. For this exercise, well use the defaults provided by AFW. Click *Finish* to execute the add process. The form is added to your Visual Basic project window and a dialog confirming your successful add is displayed.

Open the form now in Visual Basic. You can see that its an exact duplicate of the original.

This is how easy it is to work with Form Templates. By now youre probably already thinking of many ideas for Form Templates of your own.

Next, lets create a new form and add the Control and Menu Templates that we captured earlier to it.

Adding the Controls to a Form

Step 8

For this exercise, add a new form to your active Visual Basic project, save it (using any name you choose), and make sure it has the input focus. Run the Control Wizard by selecting *AppFramework* + *Add* + *Controls* from the Visual Basic *Add-ins* menu.

As in Step 7, expand the Test Project Framework in the TreeView on the *Add Controls* - *Step 1* dialog to display the Controls Collections included within.   Open the Collection called OK, Cancel Collection to view the individual Templates that make-up the Collection.

The Control Wizard lets you add either an entire Controls Collection or an individual Control Template to your form. To add an individual Template, select it from within the TreeView and proceed.   In this case, however, well add the entire Controls Collection.

Select the OK, Cancel Collection in the TreeView and click *Next*.   The *Add Controls* - *Step 2* dialog itemizes the Control Templates to be added.   Click *Finish* to execute the add.   An *Add Controls* status dialog confirms the successful add.   The new controls now appear on your form in exactly the same relationship as they appeared on the form from which they were captured.   Check the code window of your form to confirm that the methods for these controls are also present.

Leave the active form open so we can add a collection of menus to it next.

Adding the Menus to a Form

Step 9

Make sure that your new form has the input focus and run the Menu Wizard by selecting *AppFramework + Add + Menus* from the Visual Basic *Add-ins* menu. In the TreeView on the *Add Menus - Step 1* dialog, open the Test Project Framework to display the available Menus Collections. Open the File Menus Collection to view the individual Menu Templates that make-up this collection. To add a single Template, you would select it and proceed to the finish. For this exercise, however, well add the entire collection of menus. Select the File Menus Collection in the TreeView and click *Next*. The *Add Menus - Step 2* dialog confirms the list of menus to be added. Click *Finish* to execute the add and *OK* in the *Add Menus* status dialog to return to Visual Basic .

As you can see, its easy to create and use all kinds of Templates with AFW. Your Libraries grow as you design more and more Templates and add them to your Library. Each new task, in turn, becomes that much easier. In addition, you can start with the Libraries shipped with AFW. Its not hard to see why AFW actually grows in value the more you use it!

Capturing a Style

Default Styles

Before we begin the process of capturing and applying a Style, its helpful to know something about *Default Styles*. Default Styles are collections of properties associated with each form and control class within a Library. When you capture a Style from an object, AFW refers to the Default Style for the applicable object class to determine which properties to include in the Style. Without Default Styles, youd have to itemize all the properties you want each time you capture a Style.

As you capture Styles, however, AFW lets you add or remove properties from the Default Styles collection before finishing the capture.   In this way, you can specify exactly which properties to include in each Style you create. You can create multiple Styles for each object class and each of these can include different sets of properties, if you want.

When you create new Libraries or use the ones shipped with AFW, Default Styles for all standard object classes in Visual Basic are already assigned. You can easily change these using the Library Browser. For more discussion of Default Styles, refer to *the Topic* - *Templates and Styles*.

A CommandButton Style

Step 10

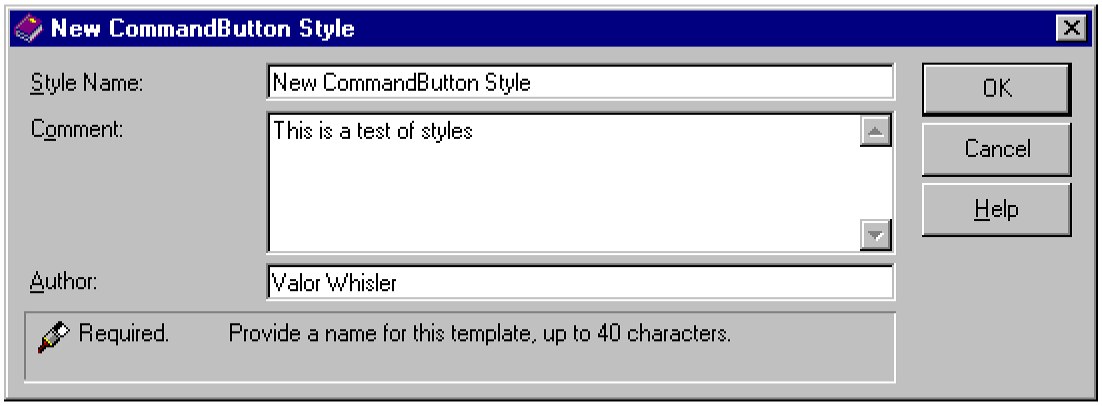
Before we can capture a Style, we first need to create or select an object whose properties are set as we want them to appear in the Style.

On your Form1, lets make sure that one of the command buttons, the OK button, has the desired properties. For a Windows 95 look, set the properties as follows: *Height* - 405, *Width* - 1125, *Caption* - OK, *Default* - True, and Name - cmdOK.

The next few steps will save a set of this buttons properties, including its *Caption*, *Left*, and *Top* properties, as a Style.

Select the OK button. Start the Capture Wizard by selecting *AppFramework* + *Capture* + *Style* from the Visual Basic *Add-Ins* menu. The *Capture Control Style - Step 1* dialog appears. Click *New*. A *New <Control Class> Style* dialog like that in the figure below appears.

Note Styles are not associated with a particular Project Framework; rather, they are global to a Library. Consequently, you dont need to specify a Project Name in the editor when working with Styles.


New CommandButton Style

Style Name: New CommandButton Style

Comment: This is a test of styles

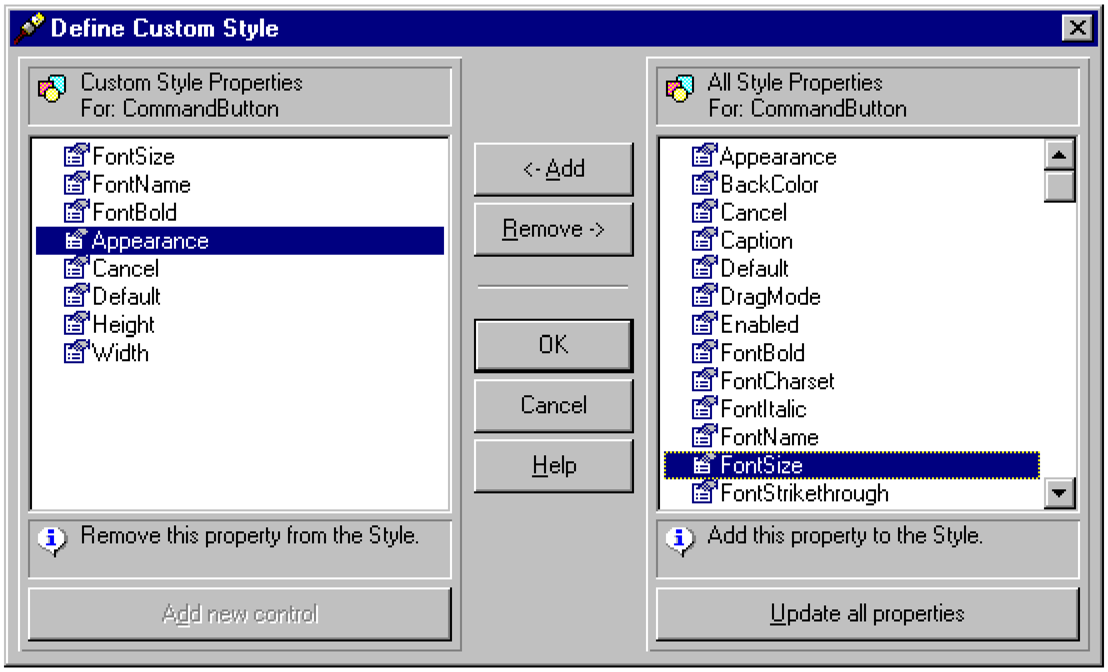
Author: Valor Whisler

Required. Provide a name for this template, up to 40 characters.

Naming the CommandButton Style

Accept the default name here and add any comments you desire. Specify yourself as the author and click *OK*. The Wizard takes you to the *Capture Control Style - Step 2* dialog. For this exercise, lets change the set of properties we will be capturing. Remember: if you dont change this set, you will capture the properties contained in the Default Style for this control class in this Library. Click the *Styles* button to capture a custom set of properties. The *Define Custom Style* dialog is displayed.

Select FontName in the right side list box and click the <- *Add* button. Repeat this for FontSize and FontBold. The list of properties youll be capturing appears in the left list box, like that shown below.



Creating a Custom Style

Click *OK* to accept the custom property set for this Style. You are returned to the *Capture Control Style - Step 2* dialog. Click *Finish*. The capture will be executed and a *Capture Status* dialog confirming the successful capture will be displayed.

Applying the Style

Step 11

Unlike Templates, which *add new objects* to your Visual Basic projects and forms, Styles are *applied* to existing objects. The following steps describe how to apply the New CommandButton Style that you just captured in Step 10 above.

First, create two more command buttons on Form1. To get the full impact of using Styles, set some unusual sizes and fonts on these buttons.

Note A Style must be captured from a single control or form, but can be applied to any number of similar controls at the same time.

Select your two new command buttons to let AFW know that you want to apply a Style to these objects. Start the Style Wizard by selecting *AppFramework* + *Apply Styles* from the Visual Basic *Add-Ins* menu. This Wizard walks you through the steps needed to apply a Style from a Library to the selected objects.

In the TreeView that appears on the *Apply Style - Step 1* dialog, you will see folders for Styles grouped by control class. Within each folder are the actual Styles listed by name. Open the *CommandButton Styles* folder by pressing the + sign to the left of the folder. Now select the folder name New CommandButton Style. Open it to view the comment and click the *Next* button.

The *Apply Style - Step 2* dialog appears, showing a list of the properties included in the selected Style and their values. Click *Finish* to apply the Style. Your command buttons will now have all of the same properties that were included in the New CommandButton Style. Remember that the only properties applied your command buttons were those included in the New CommandButton Style. No other properties were affected.

As you can imagine, Styles are a very powerful part of VB AppFramework. They make it easy to standardize the look and feel of your applications. Try experimenting with Styles of your own.

Tip Your Styles arent limited to visible properties like color, font, height and width. Try creating other categories of Styles like database Styles or help topic Styles.

This completes your first trial run with VB AppFramework. As you can see, its pretty easy. The following topics describe each of the features of AFW in more detail.

Templates and Styles

This topic defines Templates and Styles, how they relate to Visual Basic, and how they can be used.   You may be familiar with Templates and Styles from working with other products like Microsoft Word or Excel. AFW Templates and Styles are similar in concept, but actually easier to define, organize and use.

Contents

What is a Template?

A Template is a Visual Basic object definition that is stored in a Library. The term Template is used to promote the idea that these objects are intended to be generic application components that can be used over and over.

There are a couple of different approaches to designing and using Templates:

1. You can design and capture completed components from working applications. When you add them back to a Visual Basic project, theyll work just like they did before (provided, of course, that there are no unfulfilled dependencies on components external to your template).
2. You can design and capture components in near-working order, with code stubbed out to fill in the blanks and go. This approach can sometimes require more planning, but usually produces templates that are easier to reuse in more situations.

The beauty of AFW is that it lets you work either way and organize your work in whatever way best suits your needs.

Tip Placing directives in the code behind your Templates will make it easier for others to reuse your Templates efficiently.

Templates correspond directly to Visual Basic objects, with the exception of Functions. To make this clearer, lets take a look at the Visual Basic programming environment and how its objects relate to each other and the corresponding AFW Templates.

AFW provides two ways that you can work with Templates:

1. You can permanently store a Template using the Capture Wizard and apply it at any time using the applicable Add Wizards.
2. You can temporarily copy and paste Templates while working in a project using the Clone Form or Clone Controls tool. In this case, Templates are never saved into your Library. Cloning is discussed in detail in *Chapter 11 - Cloning Forms, Controls and Styles*.

VB Object	VB Container	AFW Template	AFW Container
Form	Project	Form Template	Project Template
Code Module	Project	Code Template	Project Template
Class Module	Project	Class Template	Project Template
Control	Form	Control Templates	Controls Collection
Menu	Form	Menu Template	Menus Collection
Proc	Code Window	Function Template	Library
Declaration	Code Window	Function Template	Library
Comment Block	Code Window	Function Template	Library
Properties	Control, or Form	Style	Library

Kinds of Templates

As shown in the previous table, ***Form***, ***Code*** and ***Class*** Templates correspond directly to forms, code modules, and class modules within the Visual Basic IDE. In fact, the actual text files created by Visual Basic for these components are captured into your AFW Library (the corresponding FRX files are also captured for forms). *This is why you must save your files in Visual Basic prior to performing a capture*.

Control and ***Menu*** Templates are handled differently. For these Templates, the Visual Basic object is queried and a definition of the object and its properties are stored in your Library. The methods for these Templates are extracted from the corresponding text file because the Visual Basic IDE does not expose them as separate objects through the Add-ins interface.

Functions are unique objects to AFW, but still relate to things youre familiar with in Visual Basic. A Function is any section of code found in the Visual Basic code window. It can be an entire procedure, a declaration, a comment block, or any other contiguous code segment you want to reuse. The figure below shows a code block that you might want to place at the beginning of every class module, for example.

```
'--------------------------------------------------------------
' Copyright 1995 Semiotix Systems, LLC
'--------------------------------------------------------------
' <Class> pertains to capture related functions
'------------------------------------------------

Option Explicit

'<Class>----------------------------------------------
Private Const semClassID As Long = clsidCaptureWizard
Public Application          As String
Public Name                 As String
'<EndClass>-------------------------------------------
```

A Code Block for a Class Module

What is a Style?

A Style is a collection of property settings for a Visual Basic form or control.   AFW provides a mechanism for you to automatically capture these settings from one object and re-apply them to other objects of the same class. Styles pertain only to Visual Basic forms and controls.

AFW lets you determine which properties are to be included in a particular Style; during the capture process and in the Library Browser.   Every Style can include different sets of properties if you want, or you can have multiple Styles that include the same properties but with different values.

You may be tempted to associate Styles only with visible properties like color, font and size.   In AFW, however, Styles can pertain to any collection of properties.   This makes it possible to create such Styles as Database Style or Help Topic Style.

The ability to create and apply Styles is a very powerful part of AFW.   Styles can be applied to enforce user interface consistency when working with teams of developers, with multiple projects, or with forms and controls within the same project.   You can apply Styles as you first add objects to a project or later to clean up non-conforming or incomplete work.

AFW provides two ways that you can work with Styles:

1. You can permanently store a Style using the Capture Wizard and apply it at any time using the Style Wizard.
2. You can temporarily copy and paste Styles while working in a project using the Clone Styles tool. In this case, Styles are never saved into your Library. Cloning is discussed in detail in the topic - *Cloning Forms, Controls and Styles*.   Cloning a Style is similar to using the Format Painter in Microsoft Word and Excel.

The Default Style

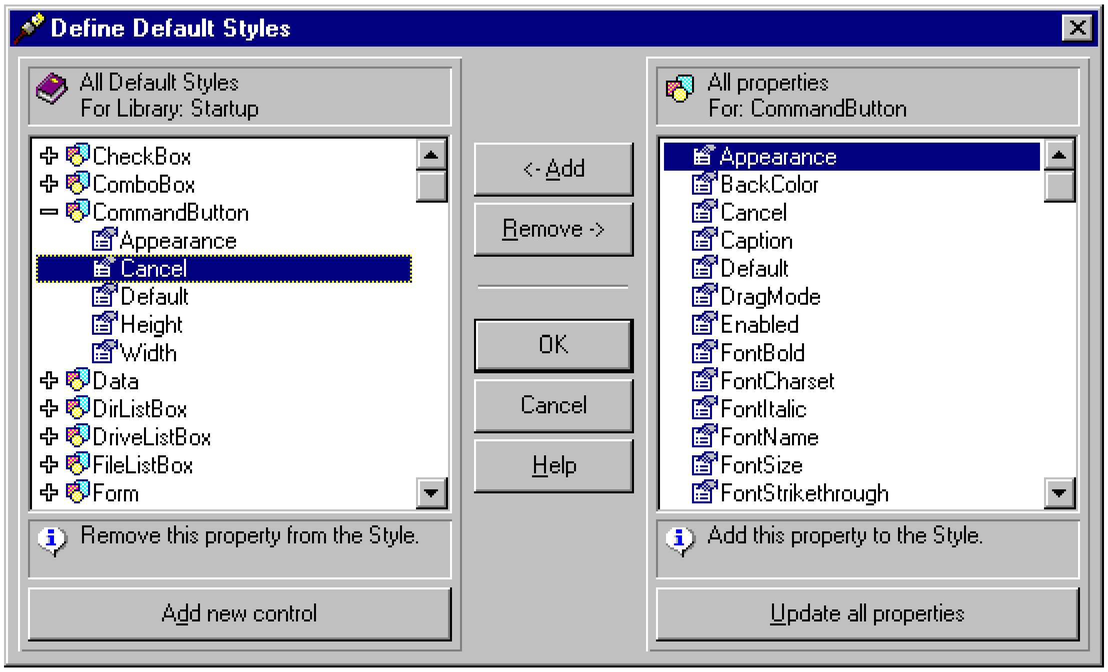
Each Style in AFW relates to a particular class of control or form (for simplicity sake, we refer to these as *object classes* below). Each Library shipped with AFW and each new Library you create is seeded with definitions for the standard object classes that ship with Visual Basic. As you capture new objects (such as third party custom controls), the attributes of those object classes are automatically added to your Library.

In addition to the initial definition of each object class, AFW also stores a collection of properties known as the *Default Style* for each class. The Default Style serves two purposes:

1. It seeds the properties collection for each new Style you capture. All Styles you capture start with the set of properties defined in the Default Style. You can add or remove properties from this set in the last step of the Capture Wizard if you want your Style to include a different set of properties.
2. It is the properties collection that is used by Clone Styles tool. When you copy and paste Styles using the Clone Styles tool, the list of properties captured is derived from the Default Style for that object class.

Note You can change the Default Styles at any time. Remember, though, that when you change Default Styles, you are only changing them for one Library at a time.

To change the Default Style, select *Library* + *Default Styles* from the menus in the Library Browser. The *Define Default Styles* dialog is displayed with two outlines. The left outline lists all object classes found in the Active Library. Expanding any object class in the list reveals its Default Style properties. The right outline lists all properties for the selected object class. You can add or remove properties to the Default Style for any class using the appropriate buttons on the form.



Defining Default Styles

Adding Controls to a Library

The *Add new control* button under the left outline of the *Define Default Styles* form allows you to add new object classes to your Library, such as 3rd party custom controls. When you add a new object class, you add the definition for that control including its required custom control reference and all of its properties.

Note This function is provided to manually add object classes to your Library. This is not a required step because new object class definitions are automatically added by AFW to the Active Library during the capture process if the captured object is new to the Library.

Updating Control Properties

The *Update all properties* button under the right outline on the *Define Default Styles* form allows you to update the definition of each object class.   This may be needed, for example, when a custom control is updated to a new version and the vendor has changed its properties in the new version.

VB AppFramework does <u>not</u> update control class definitions each time you capture a control instance. For performance reasons, this is done only the <u>first</u> time that a control class is seen by AFW.   By selecting an instance of the affected class in your Visual Basic environment and pressing *Update all properties* button, your Library will be updated with a current set of properties.   The new set of properties is reflected in the right outline.

If you notice that the set of properties listed in the right outline is incorrect or out-dated, you should perform this function.   This can happen if new versions of custom controls are installed on your system after a prior version has been captured into the Library.

The Library Browser

The Library Browser helps you manage your VB AppFramework Libraries.   You can use it to:

- Browse the contents of Libraries and select the Active Library;
- Create, install, remove, compact and repair Libraries;
- Manage Project Frameworks;
- Edit and delete Templates
- Report on Libraries

This topic discusses the AFW Library and the Library Browser; what they are and how to use them. If you want a more technical perspective of an AFW Library, see the VB AppFramework Knowledge Base help file that is placed on your system when you set up AFW.

Contents

What is a Library?

A Library is the repository for reusable Templates and Styles that you capture using AFW.   Each Library is constructed as a standard MS Access 2.0 database file.   AFW automatically creates and updates these databases, so you normally dont have to worry about their exact contents or structure.   The standard database format, however, lets you view, edit, and report on your Libraries using external applications like Microsoft Access.

Each AFW Library contains one or more Project Frameworks and collections of Styles and Function Templates.   Each Project Framework, in turn, is a collection of individual Templates, including form, control, menu, class module and code module Templates.   These are described further in the proceeding topic.

Visual Basic projects are comprised of forms, class modules, and code modules. AFW further breaks these objects into sub objects, each stored as individual entities in the Library.   A Form Template captures an entire form, including its controls and methods.   A Control Template captures an entire control, including all its properties and methods.   A class or code module Template captures all code contained in these modules.

AFW Libraries

The Startup Library

This Library is a sampler that is included with the product.   It contains some Functions, Styles, controls, and menus, as well as several Project Frameworks.

The Template Library

This Library is a database template used to create new Libraries. When you create a new Library in AFW, you are actually making a copy of the Template Library file. The Template Library is reserved and cannot be used for capturing new Templates and Styles.

Professional Libraries

Several professionally developed Libraries of Templates and Styles are available from Semiotix. Using them will increase your productivity by eliminating the need to create your own Templates for many common tasks. The MDI framework, for example, contains a collection of Templates that comprise a complete MDI application. Menus, code, child forms, tabbed property sheets; they are all available and ready to use! You can insert this entire framework into your Visual Basic application in just one step, saving many hours of work.

As many developers and companies begin working with AFW, we anticipate thatother exciting new Libraries will become available from Semiotix and other third parties.

Your Libraries

AFW makes it easy to develop and reuse your own application components and your own style of programming. For instance, you can quickly add a standard About box to your Library or your own combinations and styles of controls, such as the sizing and placement of command buttons. Now, every time you want to add a pair of OK-Cancel buttons to a form, you can let AFW reduce the task to a few mouse clicks.

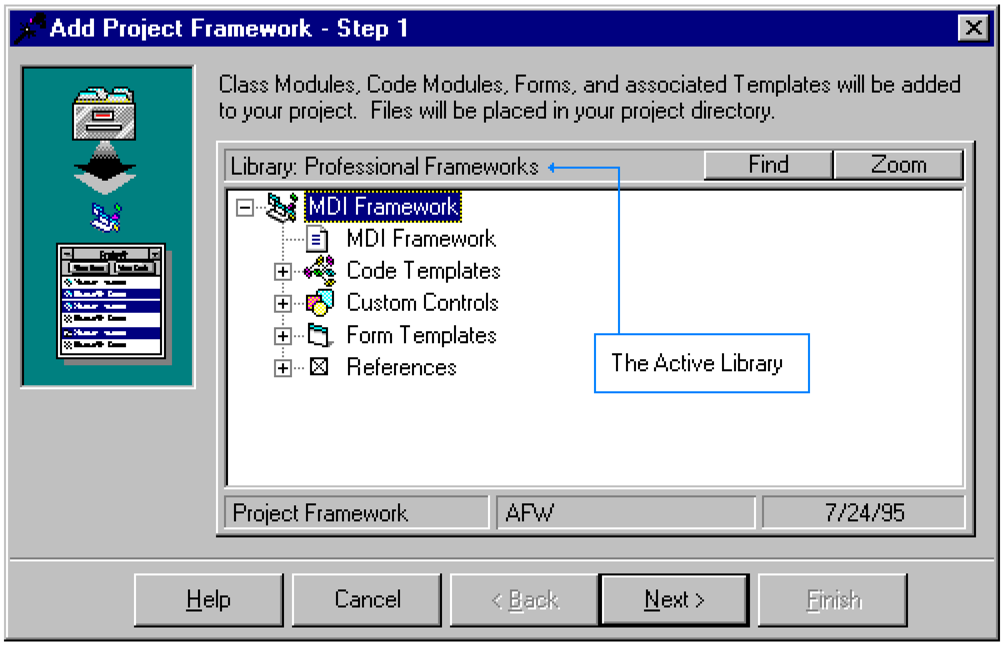
As you gain experience with AFW, youll discover many new ways to help with repetitive programming tasks. Now its easy to capture the professional look and feel of that perfect form, and apply it to all your applications!

Instead of distributing printed programming and style guidelines throughout an organization, AFW lets you send a Library to all parties involved or share a Library on a network server. Each developer can quickly add standard Templates and apply Styles to achieve a consistent, corporate-wide look, architecture and programming style among all applications. The consistent and professional touch youve been seeking for all your Visual Basic applications will finally be within reach.

The Active Library

To use a Library in AFW, you must make it the *Active Library*. This is done by highlighting the desired Library in the Library Browser and selecting *Library + Activate* from the browser menu.

For convenient reference, the Active Library is noted in status box above the TreeView control in *Step 1* of all the Wizards . The Active Library is also referenced at all times when working in the Library Browser.



Add Project Framework - Step 1

Class Modules, Code Modules, Forms, and associated Templates will be added to your project. Files will be placed in your project directory.

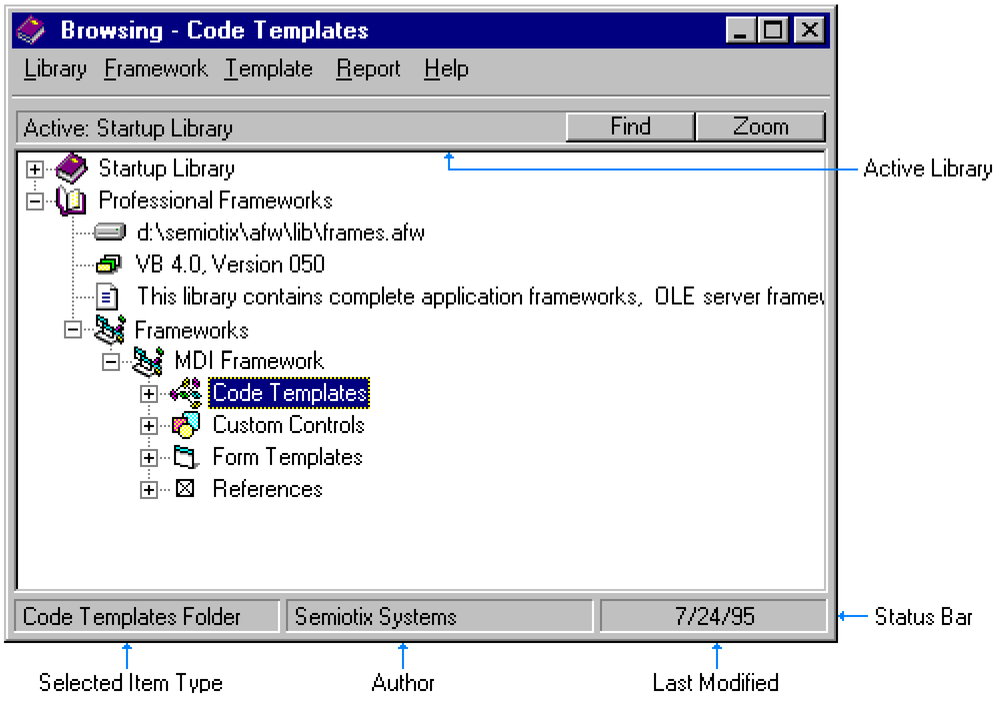
Library: Professional Frameworks

Project Framework	AFW	7/24/95

The Active Library in the Wizards

Using the Browser

Selecting *AppFramework + Library Browser* from the Visual Basic *Add-ins* menu takes you to the Library Browser. This form is used to maintain AFW Libraries. The following figure shows some of the features of the Browser.



Active: Startup Library

Startup Library
Professional Frameworks
d:\semiotix\afw\lib\frames.afw
VB 4.0, Version 050
This library contains complete application frameworks, OLE server framev
Frameworks
MDI Framework
Code Templates
Custom Controls
Form Templates
References

Code Templates Folder | Semiotix Systems | 7/24/95

The Library Browser

To view items in any Library, simply expand the TreeView to the desired level of detail. You may notice a slight delay as this expansion takes place. This is a result of AFW going out and querying the Library database.

Installing a Library

A Library must be installed in the browser before it can be used. This feature is provided so that Libraries which have been developed somewhere elseperhaps by another development teamcan be used. To install a Library, select *Library* + *Install* from the Library Browser menus, navigate to the desired Library file, and click *OK*. The Library remains in the list until it is removed. You can have up to 20 Libraries installed at any one time.

Removing a Library

If you no longer want to use a particular Library, you can remove it.   This does NOT delete the Library database file from your system, but merely removes a reference to it in the AFW Library Browser.

To remove a Library from the Browser, select it in the TreeView and choose *Library* + *Remove* from the Library Browser menus. If you want to delete the Library database file from your system, you can use the standard tools or utilities provided with your operating system.

Compacting and Repairing Libraries

Its a good idea to compact your Libraries every so often to free up unused space. This is done by selecting the Library and choosing *Library* + *Compact* from the menus. You can compact any Library, including the active one. AFW will close the Active Library and then re-open it after compacting.

If you receive errors when using a Library, you may need to repair it. This is done by selecting the Library in the TreeView, and choosing *Library* + *Repair* from the Library Browser menus. It is good practice to compact a Library once it has been repaired.

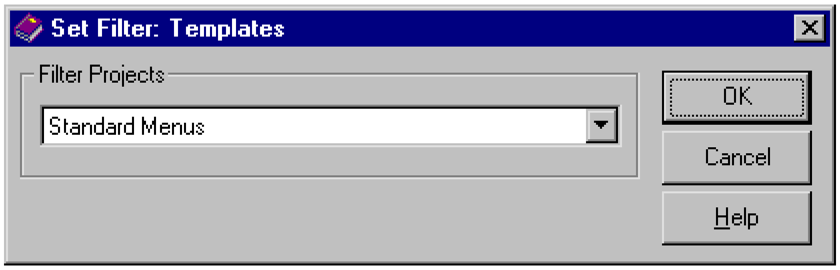
Reporting

AFW ships with six pre-formatted reports. These reports provide another way to view the data contained within your Libraries. They are also useful for auditing and documenting Libraries.

Reporting on your Libraries is either a two or three step process:

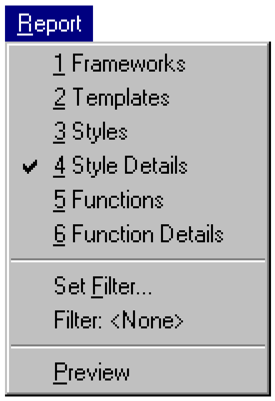
1. Select the desired report by checking it on the menu.
2. Set a filter (optional).
3. Preview the report.

To choose the report to run, select one of the menu items *1-6* located under the *Reports* menu in the Library Browser. If you want to filter the contents of a report, select the menu item *Report* + *Set Filter*. Note that the currently set filter is listed directly below the *Set Filter* menu item. The filter choices will vary, depending on the report that is selected. The figure below shows a filter being set to report only on a particular Project Framework. If the first item in the combo box is <None>, then no filter will be used (i.e. in this case, all Project Frameworks will be included in the report).



Setting a Project Filter

Once a filter has been set, it will appear in the menu item directly below *the Set Filter* item.



The Current Filter

Select the menu item *Report* + *Preview* to run the selected report. Once in Crystal Reports, you will have a number of reporting options available such as printing the report or exporting it to various formats and destinations. Refer to the Crystal Reports documentation included in Visual Basic for complete descriptions of these features.

Note AFWs built-in reports were developed with the Crystal Report designer that comes with Visual Basic. The individual report templates can be found in the *Reports* directory under the directory where AFW is installed.

Capturing Templates and Styles

The Capture Wizards allow you to populate Libraries with Templates and Styles of your own design. These wizards are accessible through the AFW *Capture* menus.   The process of creating Templates and Styles is always done within Visual Basic.   Once youve designed (or selected) an object that will serve as the basis for a Template or Style, the Capture Wizard provides a step-by-step approach for getting it into a Library.

The Capture Wizard lets you create new Templates and Styles or replace existing ones. If you need to edit information contained in an existing Template, such as a comment or author name, or if you need to delete a Template, you must do so in the Library Browser.

This topic describes the capture process in general.   If you want to try a working exercise of this Wizard in action, see the topic - *Your First AFW Library*.

Contents

Overview of the Capture Process

The AFW Capture Wizards interact with the Visual Basic IDE to capture definitions of Visual Basic objects (Templates) and collections of their property settings (Styles) into your Active Library. They do not alter your original Visual Basic projects or the objects contained in them in any way.

Once captured, any of your Styles and Templates can be used in any Visual Basic project, anytime, simply by running the applicable A*dd* or A*pply* Wizard.

The general process for capturing Templates and Styles is the same, regardless of what you are capturing:

1. Design the object(s) that will serve as the basis for your Template or Style in Visual Basic, or open a project that contains the object(s) you need.
2. Select the Visual Basic object(s) you want to capture.
3. Start the appropriate Wizard from the AFW *Capture* or *Associate* menus.
4. Click *New*, or highlight the Template or Style to replace and click *Replace*.
5. Provide a name for the Template or Style and add comments and an author name, if you want.
6. Click *Finish* to complete the capture.
7. Depending on the type of object you are capturing, there may be a few extra steps that well discuss below, but the process remains basically as outlined above. Its that simple. There are no cumbersome file management, directory organization or other awkward processes to follow, and you wont need special training to be productive with AFW right out of the box.

Capturing Forms

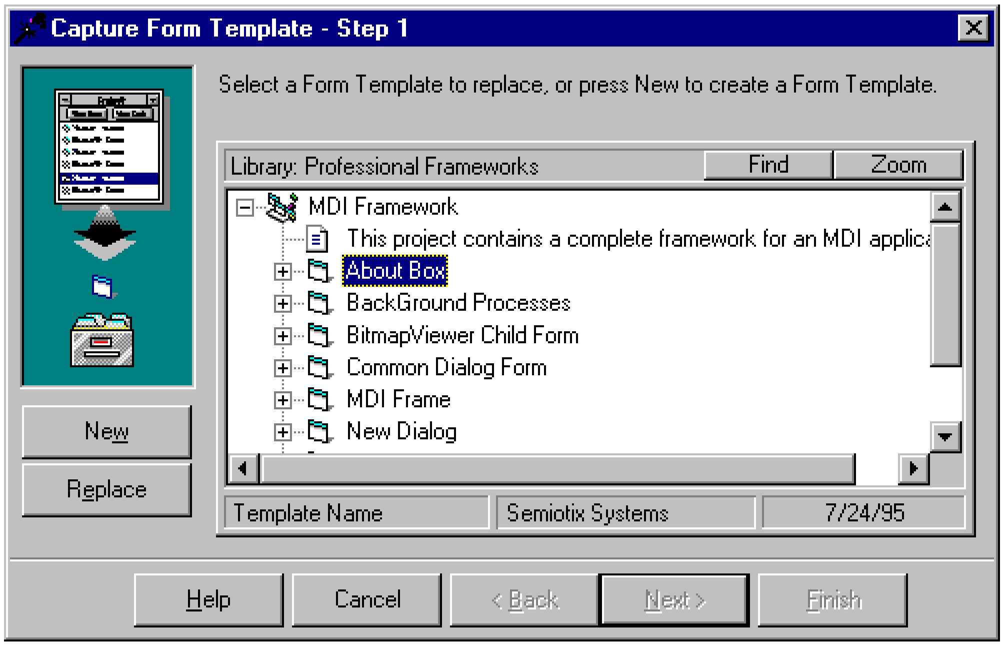
This section discusses the process of capturing a Form Template.

As in any capture process, you must have the desired object selected in the Visual Basic design environment before proceeding.   In this case, the object will be a Visual Basic form.

To capture a form, proceed as follows:

Select a form in your active Visual Basic project, then Select *AppFramework* + *Capture* + *Form* from the Visual Basic *Add-ins* menu.   The *Capture Form Template - Step 1* dialog will be displayed.

At this point, VB AppFramework will query the Visual Basic IDE to determine what type of object is selected.   If the type of object you selected isnt consistent with your AFW menu selection, youll get an error message to this effect and will have to restart the capture process.


Capture Form Template - Step 1

Select a Form Template to replace, or press New to create a Form Template.

Library: Professional Frameworks

- MDI Framework
  - This project contains a complete framework for an MDI applica
  - About Box
  - BackGround Processes
  - BitmapViewer Child Form
  - Common Dialog Form
  - MDI Frame
  - New Dialog

Template Name	Semiotix Systems	7/24/95

Capturing a Form Template

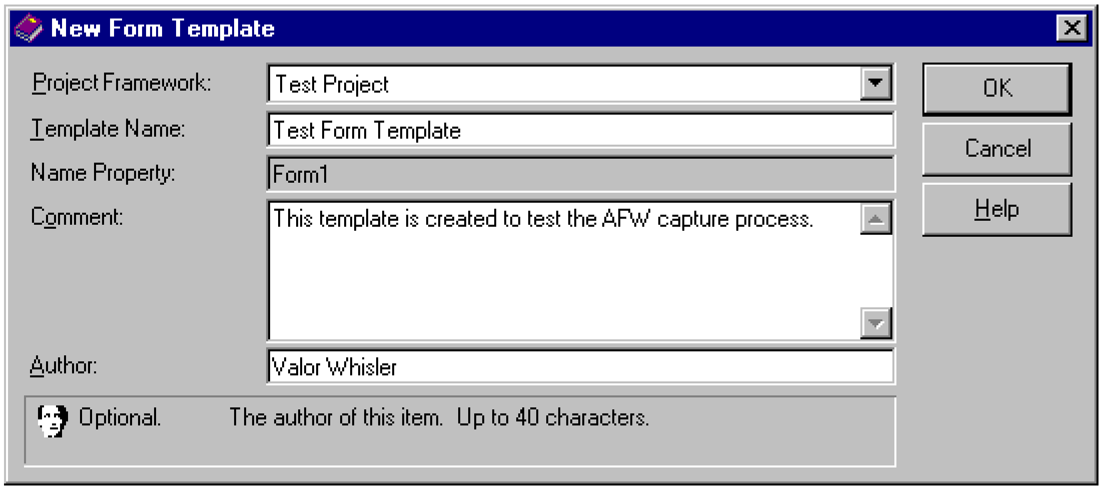
The *Step 1* dialog gives you the option of creating a new Template or replacing an existing one, if any are listed.   To create a new Template, click *New*.   To replace an existing Template, select it in the outline and click *Replace*.

The *New/Replace Form Template* dialog will now be displayed. Its data fields will be seeded with the following information:

- Project Name - if you are creating a new Template, the Framework you navigated to in the *Step 1* dialog or the first Framework listed in the TreeView if you selected nothing; *or* if you are replacing a Template, the parent Framework of the Template you selected for replacement.
- Template Name - if you are creating a new Template, a concatenation of the Name property of the selected form and the word Template; *or* if you are replacing a Template, the existing name of the Template being replaced.

- Name Property - the Name property of the selected form, as obtained from Visual Basic (this is not editable).
- Comment - if you are creating a new Template, nothing; *or* if you are replacing a Template, the comment (if any) previously specified for that Template.
- Author - if you are creating a new Template, the author name from your most recent capture or from the Library if you have not captured anything in this session; *or* if you are replacing a Template, the author name (if any) specified previously for that Template.

As you move from field to field, the hints area on the bottom of the form tells you whether each field is optional or required and provides other information about the field.



Project Framework: Test Project
Template Name: Test Form Template
Name Property: Form1
Comment: This template is created to test the AFW capture process.
Author: Valor Whisler
Optional. The author of this item. Up to 40 characters.

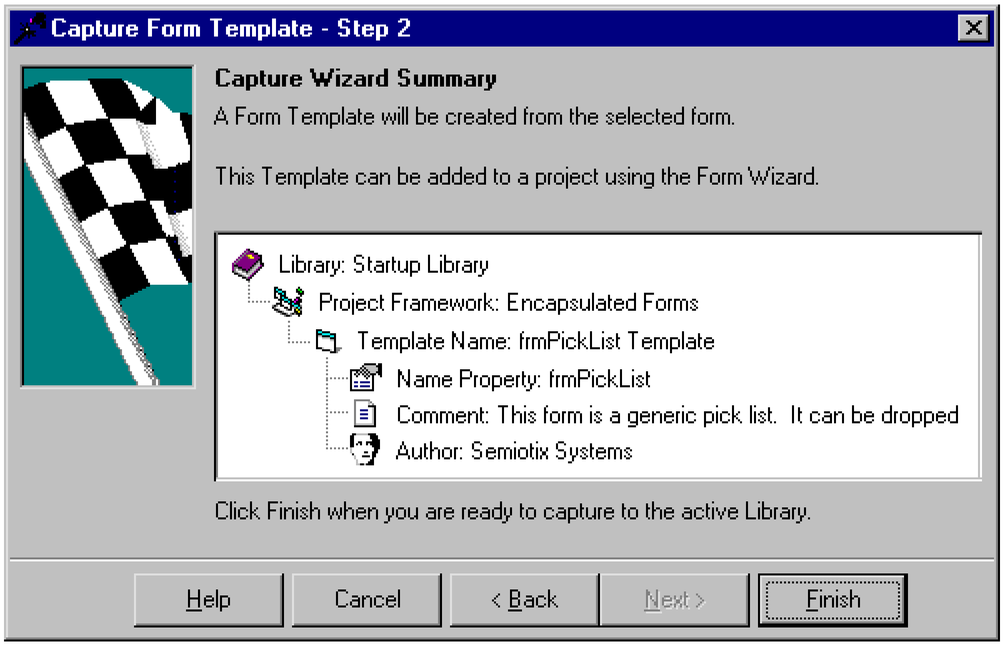
Capture Form Template Editor

To select the Project Framework with which this Template will be associated, use the *Framework Name* combo box. If you want to capture this Template under a new Project Framework that is not listed, you must cancel out of the current capture, add the Project Framework in the Library Browser, and restart the capture. You must complete all required information before clicking *OK* to proceed to the *Capture Form Template - Step 2* dialog.

Note You cannot change the Name Property of a Form, Code, or Class Template in the *New <object type>Template* dialog. The value of Name Property is the object name as it appears in your Visual Basic project.

The *Step 2* dialog recaps your Form Template name, comments, and other information. You cannot edit the information here. If you need to edit the data, you can click *Back* to return to the previous step. Click *Finish* to complete the capture. Your Visual Basic form file and FRX file (if any) will be added in their entirety to the Active Library. Your Form Template is now permanently stored and ready to use at any time using the Form Wizard.

Note If the form you wish to capture has custom controls on it, you must add the applicable references to your project before the Form Template can be added.


Capture Form Template - Step 2

Capture Wizard Summary

A Form Template will be created from the selected form.

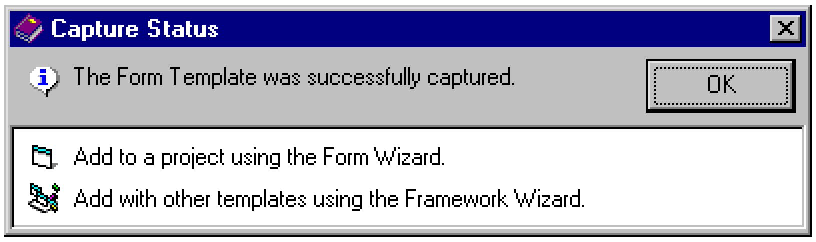
This Template can be added to a project using the Form Wizard.

Library: Startup Library
Project Framework: Encapsulated Forms
Template Name: frmPickList Template
Name Property: frmPickList
Comment: This form is a generic pick list. It can be dropped
Author: Semiotix Systems

Click Finish when you are ready to capture to the active Library.

Capture Form Summary and Finish

Once a successful capture is complete, the *Capture Status* dialog shown in the next figure will be displayed. In this dialog, the applicable Library, Project Framework and Template names are noted along with a reminder that this Template can be added back to any Visual Basic project using either the Form Wizard (to add the Form Template individually) or the Framework Wizard (to add the Form Template as part of an entire Project Framework).


Capture Status

The Form Template was successfully captured.

Add to a project using the Form Wizard.
Add with other templates using the Framework Wizard.

A Successful Capture

Capturing Class Modules

This section discusses the process of capturing class modules as Class Templates. This process is identical to that for capturing forms, except that you would first highlight a class module in your Visual Basic project window and then select *AppFramework* + *Capture* + *Class Module* from the Visual Basic Add-ins menu. Refer to the *Capturing Forms* section above for complete details on this process.

Capturing Code Modules

This section discusses the process of capturing code modules as Code Templates. This process is identical to that for capturing forms, except that you would first highlight a code module in your Visual Basic project window and then select *AppFramework* + *Capture* + *Code Module* from the Visual Basic Add-ins menu. Refer to the *Capturing Forms* section above for complete details on this process.

Capturing Controls

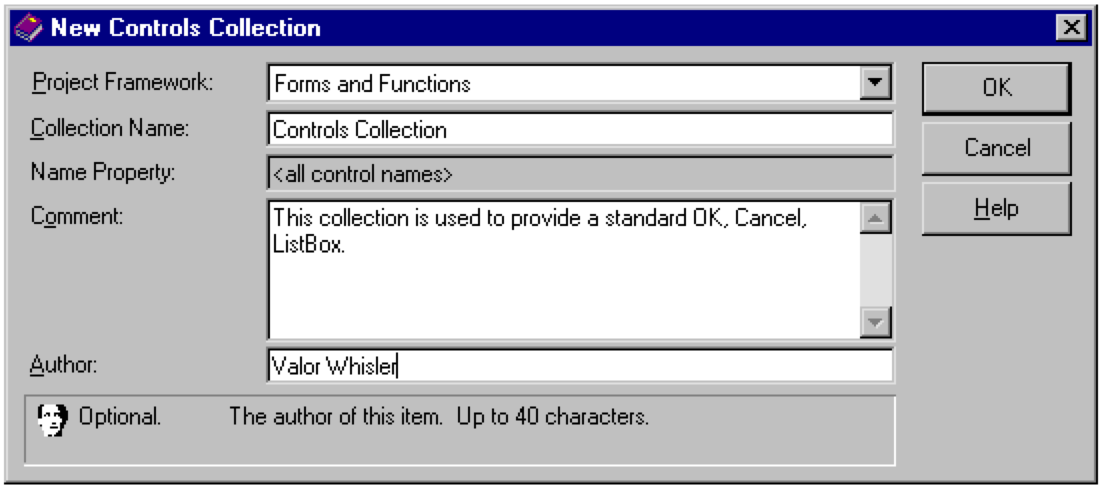
AFW lets you capture a single control or logically-related group of controls using the Capture Wizard. All controls you capture in AFW are stored as a *collection*. When you capture a single control, you are simply capturing a collection of one.

When you capture controls, you are capturing their definition in Visual Basic, including any required custom control references, all their property settings and all methods relating to them. Furthermore, all controls in a collection maintain their relative positions and relationships to each other, just as they appeared on the Visual Basic form from which they were captured.

To capture controls, proceed as follows:

1. Within Visual Basic, highlight a control or group of controls on a form in design mode and start the Capture Wizard by selecting *AppFramework* + *Capture* + *Controls* from the Visual Basic *Add-ins* menu. The *Capture Controls Template - Step 1* dialog will be displayed.
2. Click *New* for a new collection, or highlight an existing collection in the TreeView and click *Replace*. The *New/Replace Controls Collection* dialog shown in the figure below will now be displayed.
3. This dialog allows you to associate the controls collection with a particular project, and optionally to provide a comment and author. At this point, AFW doesnt know exactly which controls are selected because the selected controls might be containers for other controls. For this reason, the *Name Property* field shows that <all control names> will be captured. Complete the required and optional information, then click *OK*. The *Capture Control Templates - Step 2* dialog will be displayed.

Click *Finish*. The *Capture Status* dialog confirming your successful capture and reminding you how you can use your controls collection is displayed. Click *OK* to return to your work in Visual Basic.

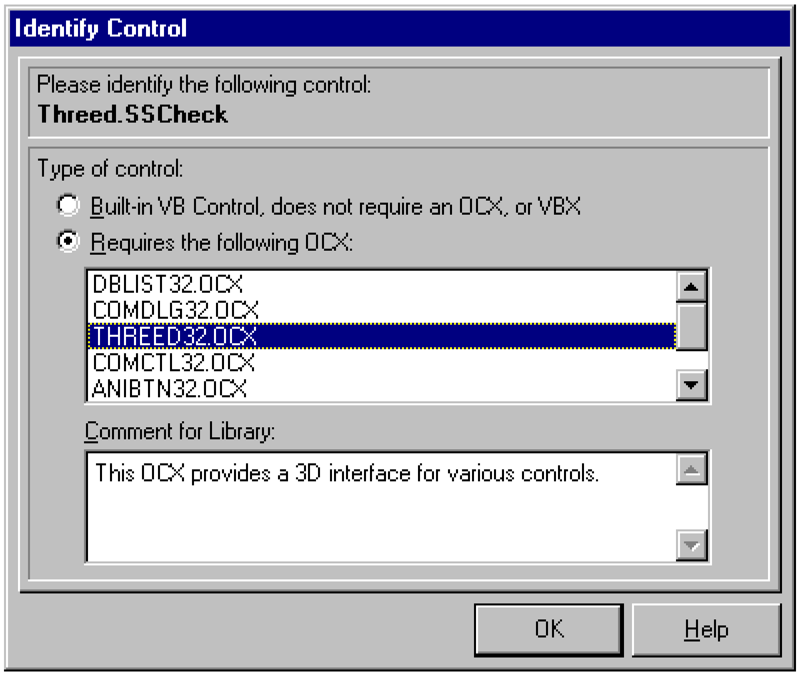


Project Framework: Forms and Functions
Collection Name: Controls Collection
Name Property: <all control names>
Comment: This collection is used to provide a standard OK, Cancel, ListBox.
Author: Valor Whisler
Optional. The author of this item. Up to 40 characters.

New Controls Dialog

Note Due to a limitation in the Visual Basic IDE, methods for the controls you are capturing must be retrieved from the physical file associated their parent form. For this reason, you must save the form before you capture controls. *Also:* some controls that contain collections of controls such as a the Tab control or TreeView cannot be captured properly.

During the capture process, AFW may ask you to identify a custom control. This request is made when AFW finds a control class that hasnt yet been defined in the Active Library. When you see the *Identify Control* dialog select the name of the VBX, or OCX that is required to use this control in your project. This step is required only once for each Library in which the control is captured.


Identify Control

Please identify the following control:
Threed.SSCheck

Type of control:

( ) Built-in VB Control, does not require an OCX, or VBX
(●) Requires the following OCX:

DBLIST32.OCX
COMDLG32.OCX
THREED32.OCX
COMCTL32.OCX
ANIBTN32.OCX

Comment for Library:

This OCX provides a 3D interface for various controls.

Identify Control Dialog

Note When you add a control to a VB form that requires a VBX/OCX Reference, AFW automatically adds the Custom Control Reference to your project before it adds the control.

Capturing Menus

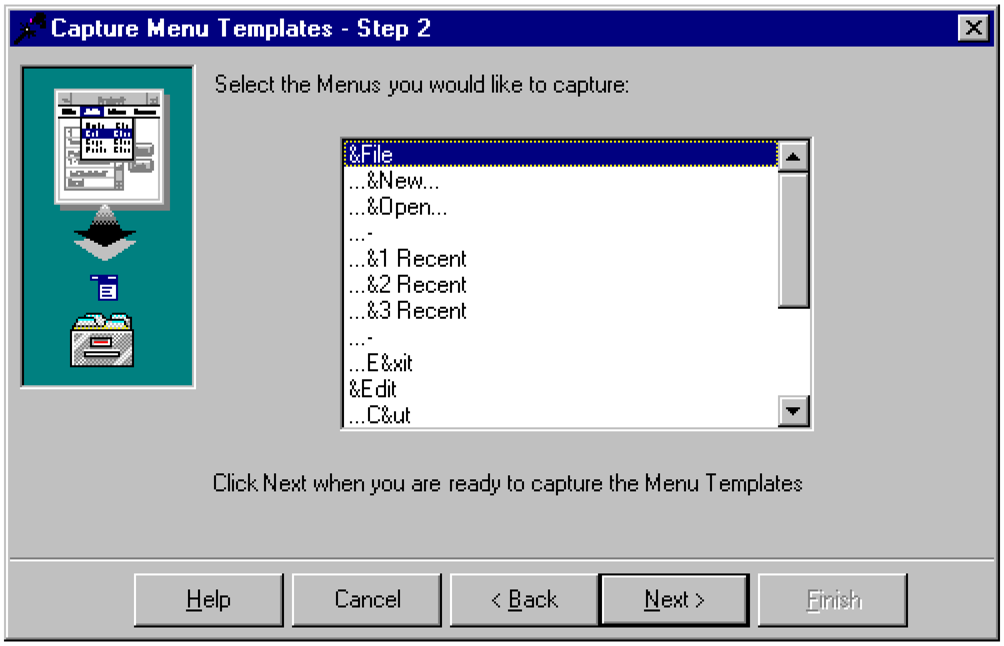
You can capture a single menu or a group of menus using the Capture Wizard. All captured menus are stored together by AFW as a *collection*. When you capture a menu, you are capturing the definition of the menu including all of its property settings, its methods, and the relationship of each menu to others in the collection.

Unlike other Templates, Visual Basic provides no way for you to select menus on a form in design time, so AFW includes a menu selection screen in the Capture Wizard.

To capture menus, proceed as follows:

1. Open a form containing the menus you wish to capture in design time and make sure the form has the input focus.
2. Start the Capture Wizard by selecting *AppFramework* + *Capture* + *Menus* from the VB *Add-ins* menu. *The Capture Menu Templates - Step 1* dialog will be displayed.
3. Click *New* to create a new menu collection, or navigate to the applicable Framework and highlight an existing menu collection and click *Replace*. The *New/Replace Menu Templates* dialog will now be displayed.
4. Enter the required Framework Name and Template Name, any optional information, and click *OK*. The *Capture Menu Templates - Step 2* dialog will now be displayed. This is where you select the menus to be captured.
5. The outline in the *Step 2* dialog lets you select multiple items. Select all the menus that you would like to capture into the current collection, then click *Next.* The Capture Menu Templates - Step 3 dialog will be displayed. Click *Finish* to complete the capture. The *Capture Status* dialog confirming your successful capture and reminding you how to use your Menu Templates is displayed.

Click *OK* to return to your work in Visual Basic.


Capture Menu Templates - Step 2

Select the Menus you would like to capture:

```
&File
...&New...
...&Open...
...-
...&1 Recent
...&2 Recent
...&3 Recent
...-
...E&xit
&Edit
...C&ut
```

Click Next when you are ready to capture the Menu Templates

Select Menus to Capture

Note You cannot select a sub-menu without selecting its corresponding parent and multiple menu selections must be contiguous.

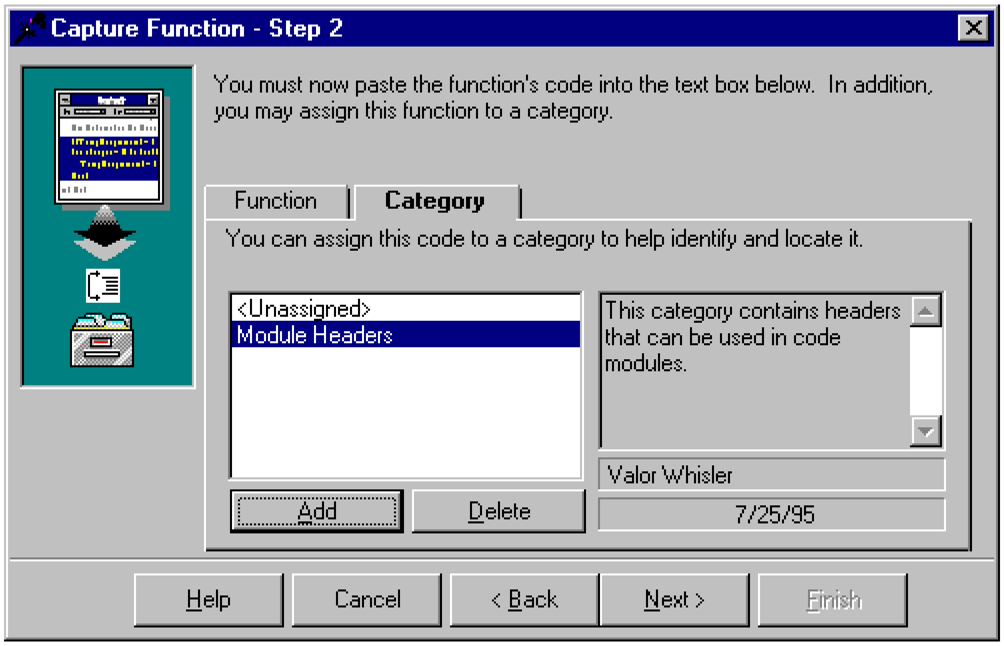
Capturing Functions

A Function Template in AFW is any piece of code obtained from a Visual Basic code window. This could be an entire procedure, a declaration or group of declarations, a comment block, or any other contiguous code segment you want to place in a Library for reuse.

To capture a Function, proceed as follows:

1. Navigate to the desired code block in your Visual Basic environment, select the block and copy it to the clipboard.
2. Bring up the Capture Wizard by selecting *AppFramework* + *Capture* + *Function* from the Visual Basic *Add-ins* menu. The *Capture Function - Step 1* dialog will be displayed.
3. To create a new Function, Click *New.* To replace an existing Function, navigate to an existing Function in the TreeView, highlight it and click *Replace*. The *New/Replace Function* dialog is displayed.
4. Enter a Template Name, fill in the Comment and Author fields if you want, and click *OK*. The *Capture Function - Step 2* dialog will be displayed.
5. Make sure that you are tabbed to the *Function* page of the dialog. You can see that AFW has automatically pasted the code you just copied to the clipboard into the text box. At this point, you can edit the code if you want before saving it into the Library. You can also paste additional text into the window using the *Paste* button located above the text box. This feature is provided so you can optionally copy and paste in additional text from Notepad or another source outside of Visual Basic before saving your Function. (Be careful not to click on *Paste* before refreshing the clipboard contents from another source. If you do so, youll duplicate the same text in your Function.
6. Tab to the *Category* page of the dialog to assign the Function to a category. Select an existing category under which you want to group this Function or click *Add* to create a new category.
7. You can also delete a Category in this dialog. If the Category contains existing Functions, you will receive a warning that deleting the Category will cause all of the Functions contained within it to also be deleted from the Active Library and be given the option of canceling the delete. You should therefore exercise *extreme caution* when deleting Categories which contain existing Functions in them. This is particularly true when you are sharing a Library with others.

Click *Next* to proceed to the *Capture Function - Step 3* dialog, then click *Finish* to complete the Function capture. A Capture Status dialog is displayed confirming your successful capture and reminding you that you can use the Function Wizard to add Functions back to any Visual Basic code window. Click *OK* to return to your work in Visual Basic.


Capture Function - Step 2

You must now paste the function's code into the text box below. In addition, you may assign this function to a category.

You can assign this code to a category to help identify and locate it.

<Unassigned>
Module Headers

This category contains headers that can be used in code modules.

Valor Whisler

7/25/95

Assigning a Function to a Category

Tip Youll probably accumulate many Function Templates as you work with AFW. Grouping Functions into categories makes them easier to find when you need to add them to projects.

Capturing Styles

To capture a Style, you must first design a Visual Basic form or control with the desired property settings. Once youve done so, you can capture these settings as a Style.

Note A Style pertains to a particular class of control or form. You can only select a single Visual Basic object to use as the Style example. Styles can, however, be applied to multiple objects of the same class.

To capture a Style, proceed as follows:

1. Select the form or control in Visual Basic which will serve as the basis for your Style. The object class noted in the Visual Basic property window is object class for which AFW assumes you are capturing the current Style.
2. Start the Capture Wizard by selecting *AppFramework* + *Capture* + *Style* from the Visual Basic *Add-ins* menu. As in Visual Basic, if you have not selected a control, then AFW assumes that you meant to select the form. The *Capture Form/Control Style - Step 1* dialog is displayed.
3. Click *New* for a new Style or highlight an existing Style in the TreeView and click *Replace*. The *New/Replace <object class> Style* dialog is displayed. Enter a Style Name and the optional Comment and Author data, if you want. Click *OK*. The *Capture Form/Control Style - Step 2* dialog is displayed.
4. The Step 2 dialog for capturing Styles is almost identical to that for capturing Templates, with one exception: an additional *Styles* button is displayed. Click this button to override the Default Style properties collection and define a custom set of properties for this Style. If you do so, the Custom Style dialog appears. The left outline displays a list of properties for this Style containing the following items:
   - If you are replacing a Style, the properties included in the Style you are replacing.
   - If you are creating a new Style, the properties included in the Default Style for the selected form or controls object class.
5. The right outline lists all properties for the selected form or controls object class. By adding or removing properties from the left outline, you can customize your Style. Click *OK* to accept your changes. You are returned to the *Step 2* dialog.
6. Click *Finish* to execute the capture. A *Capture Status* dialog is displayed to confirm your successful capture and remind you that you can apply your Style to any object(s) of the same class using the Apply Style Wizard.

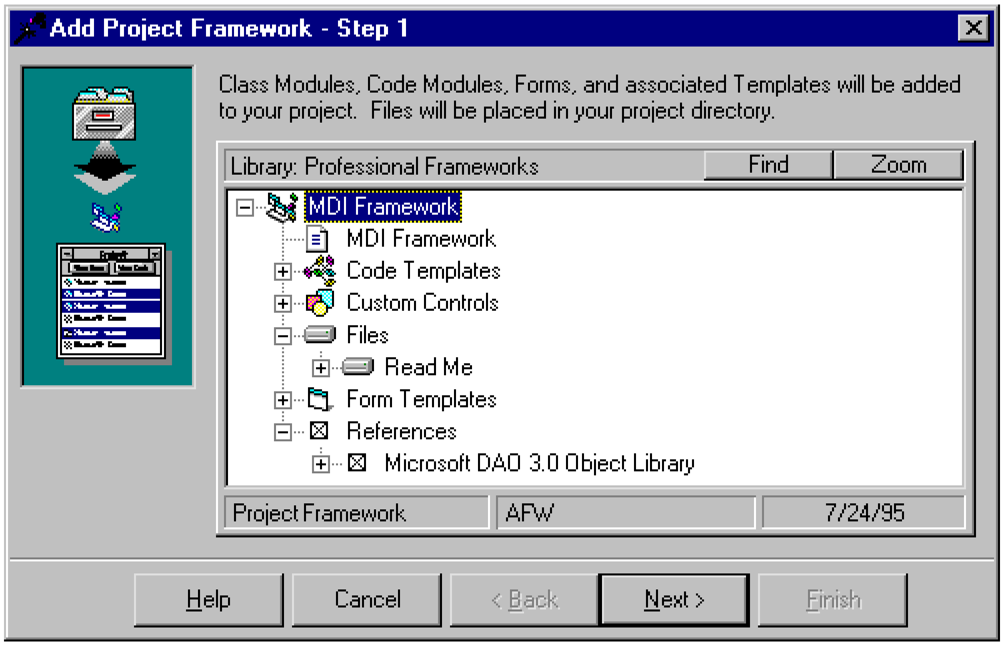
Note Picture and icon properties cannot be included as part of a Style.

Associating Items With a Project Framework

The *Associate* Wizards in AFW allow you to associate items with Project Frameworks. The reason for creating associations is to allow you to build Project Frameworks that include all custom control references, OLE object references, resource files and even external documentation files you might need to work on a project.

When you lay out a Project Framework that contains these associations into a Visual Basic project, AFW not only adds all Templates contained within the Framework to your project, but also sets the applicable custom control and OLE object references in your project, adds a resource file (if included in the Framework), and copies external files like bitmaps and technical documents to your project directly.

The process of associating items with Frameworks is very similar to that of capturing Templates.


Add Project Framework - Step 1
Library: Professional Frameworks
Find
Zoom
Help
Cancel
< Back
Next >
Finish

Class Modules, Code Modules, Forms, and associated Templates will be added to your project. Files will be placed in your project directory.

```
MDI Framework
    MDI Framework
    Code Templates
    Custom Controls
    Files
        Read Me
    Form Templates
    References
        Microsoft DAO 3.0 Object Library
```

Project Framework	AFW	7/24/95

Items Associated with a Project

Contents

Associating Custom Controls

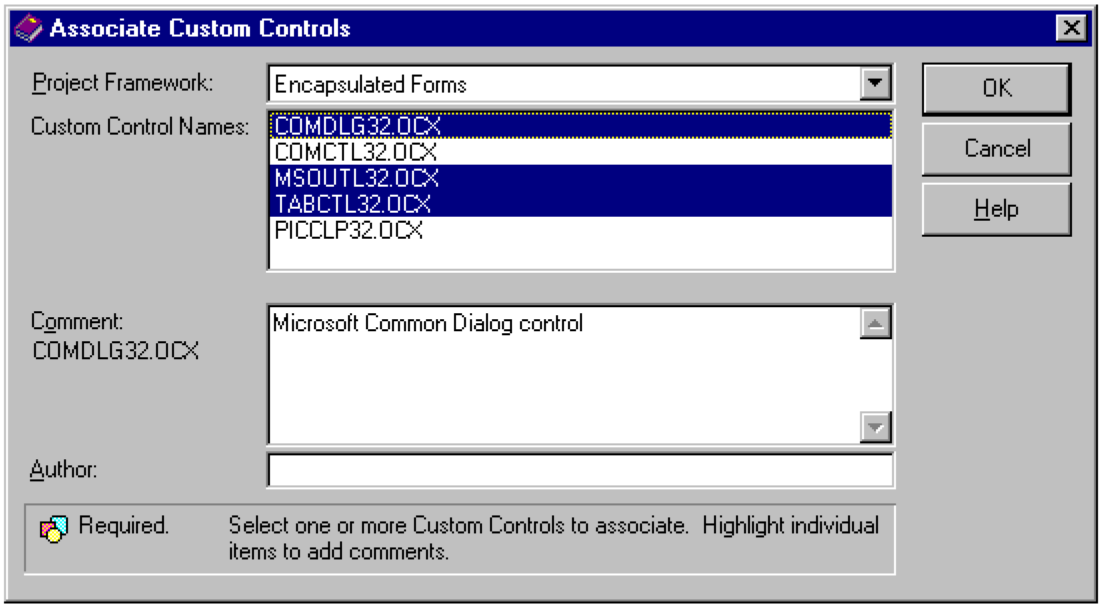
To use custom controls in a Visual Basic project, you must first reference them in the project. This is normally done in *Custom Controls* dialog that is accessed by selecting *Tools* + *Custom Controls* from the Visual Basic menu.

To make working with Project Frameworks easier, AFW lets you associate custom control references with them. When you lay out a Framework into a Visual Basic project, AFW will automatically set the required custom control references for you.

To associate custom controls with a Project Framework, do the following:

1. Select AppFramework + Associate + Custom Controls from the Visual Basic Add-ins menu. The Associate Custom Controls - Step 1 dialog is displayed.
2. Press New to create a new association or highlight an existing Association in the TreeView and click Replace. The New/Replace Custom Controls Association dialog is displayed.
3. Select the Project Framework with which you want to create an Association. If you are creating a new Association, then select one or more Custom Control Names from the Custom Control Names list box to be associated. You can specify a Comment and an Author for each custom control you select. If you are replacing an existing Association, you can edit the data only for the single custom control selected in Step 1. When you have finished your edits, click Next.

Click *Finish*. A Capture Status dialog appears to confirm your choices. Click *OK* to return to your work in Visual Basic.



Project Framework: Encapsulated Forms

Custom Control Names:
COMDLG32.OCX
COMCTL32.OCX
MSOUTL32.OCX
TABCTL32.OCX
PICCLP32.OCX

Comment: COMDLG32.OCX — Microsoft Common Dialog control

Author:

Required. Select one or more Custom Controls to associate. Highlight individual items to add comments.

Selecting the Controls to Associate

Note AFW allows you to associate custom controls that are currently referenced in the active Visual Basic project. This association does not violate custom control licenses because only the registration ID is stored, not the executable files.

Associating References

Visual Basic lets you incorporate OLE objects, like Excel spreadsheet components or other OLE servers you may have designed in Visual Basic or Visual C++ into your applications. To do this, however, you must first reference these objects in your Visual Basic project. This is normally done in the *References* dialog that is accessed by selecting *Tools* + *References* from the Visual Basic menu.

AFW lets you associate OLE objects with a Project Framework so that, when you lay out the Framework out in a Visual Basic project, any required OLE references are automatically set for you.

To associate an OLE object (or object library) with a Project Framework, proceed as follows:

1. Select *AppFramework* + *Associate* + *References* from the *Add-Ins* menu. The *Associate References - Step 1* dialog is displayed.
2. Select *New* to create a new association or highlight an existing Association in the TreeView and click *Replace*. The *New/Replace References Collection* dialog is displayed.
3. Select the Project Framework with which you want to create an Association. If you are creating a new Association, then select one or more Reference Names from the Reference Names list box to be associated. You can specify a Comment and an Author for each Reference you select. If you are replacing an existing Association, you can edit the data only for the single reference selected in *Step 1*. When you have finished your edits, click *Next.* The *Associate References - Step 2* dialog is displayed.
4. Click *Finish*. A Capture Status dialog appears to confirm your choices. Click *OK* to return to your work in Visual Basic.

Note AFW allows you to associate References that are listed in the active Visual Basic project. This association does not violate OLE Server application licenses because only the registration ID is stored in the Library, not the executables.

Associating a Resource File

Visual Basic Version 4.0 now lets you externalize strings, bitmaps and other elements from your application in a Resource File. This makes internationalization and application maintenance easier. Before you can use a Resource File in a project, however, you must add it to your Visual Basic project window. This is normally done by selecting *Add File* from the Visual Basic *File* menu and selecting a Resource File from the FileOpen dialog.

AFW lets you associate a Resource File with a Project Framework in your Library so that, when you lay out the Framework in a Visual Basic project, the Resource File is automatically copied into your project directory and added to the project window for you. Unlike associating custom controls and OLE object references, however, AFW does capture the physical Resource File into your Library. Thats because there are usually no licensing issues involved with replicating Resource Files.

To associate a Resource File with an AFW Project Framework (and physically copy the file into your Library), proceed as follows:

1. Select the Resource File to be associated in the Visual Basic project window. Because Visual Basic lets you use only one Resource File per project, you can only associate one Resource File per Project Framework in AFW.
2. Select AppFramework + Associate + Resource from the Visual Basic Add-Ins menu. The Associate Resource File - Step 1 dialog is displayed.
3. Click New to associate a new Resource File or highlight an existing Resource File in the TreeView and click Replace. The New/Replace Resource File dialog is displayed.
4. Select the Project Framework with which you want to create an association and enter a Resource Name and a Comment and Author, if you want. Click Next. The Associate Resource File - Step 2 dialog is displayed.
5. Click Finish to execute the association. A Capture Status dialog appears to confirm your successful capture. Click OK to return to your work in Visual Basic.

Tip Resource files are different for 16 and 32 bit environments. Provide a name or comment for the Resource File that indicates the target environment.

Associating Files

Up to now, everything weve associated with a Project Framework corresponds with an item or reference that you can manage directly within the Visual Basic IDE. Typically, however, these items are only a subset of the materials relating to a development project. Other files such as bitmaps, specification documents and readme files may also be vital to your work. AFW lets you associate these external files with your Project Frameworks and then copy them out to your Visual Basic project directory whenever they are needed.

For example, the AFW Professional Framework Library uses this capability to associate Excel worksheets and Access front-ends with the *OLE Server Framework* included in the *AFW Professional Framework* Library. When the *OLE Server Framework* is laid out, it can instantly be tested using Excel and Access because the required .XLS and .MDB files are automatically copied by AFW to the proper directory on your computer.

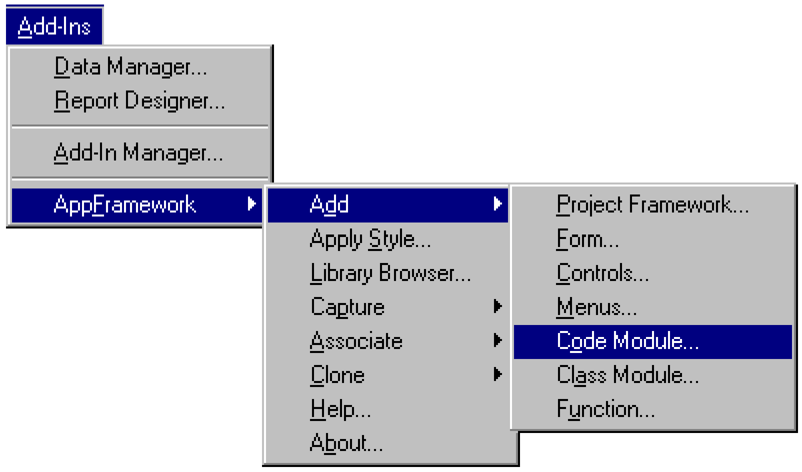
To associate an external File with any Project Framework (and in doing so, also copy the physical file into your Library database), proceed as follows:

1. Select *AppFramework + Associate + File* from the Visual Basic *Add-Ins* menu. The *Associate Files - Step 1* dialog is displayed.
2. Click *New* to create a new association or highlight an existing file in the TreeView and click *Replace*. The *New/Replace File Association* dialog is displayed.
3. Select a Project Framework with which you want to create an association. Provide a Template Name. Type in the File Name to be associated or click *Browse* to select a file using the standard FileOpen dialog. Add a Comment and Author name if you want, then click *Next*. The *Associate File - Step 2* dialog is displayed.
4. Click *Finish* to execute the association. A Capture Status dialog appears to confirm your successful capture. Click *OK* to return to your work in Visual Basic.

Note When a Project Framework is laid out, any associated Files are placed in the directory of the active Visual Basic project. Since these files are not tracked by the Visual Basic IDE, they do not appear in your project window.

Adding Form, Class, and Code Templates

This topic discusses the process of adding Form, Class, and Code Templates from an AFW Library into your active Visual Basic project. Form, Class and Code Templates correspond directly to Forms, Class Modules and Code Modules in your project window. These Templates are added using the AFW *Add* Wizards.



The Add Menus

Contents

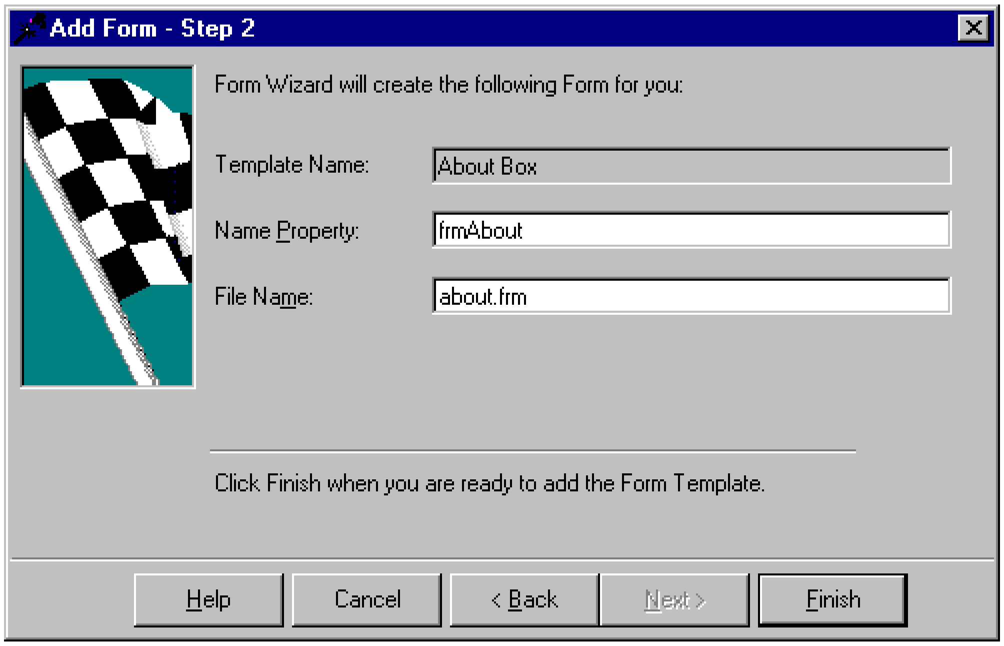
Overview

Templates are organized into Project Frameworks during the capture process. This organization serves two purposes: (1) to help group similar Templates together; and (2) to create integrated application frameworks.

Templates within a Framework can be individually added back into a Visual basic project using the applicable Add Wizards. Templates can be added as a group, along with any associated items, using the Framework Wizard.

The process of adding Form, Class, and Code Templates is identical:

1. Open the Visual Basic project to which you want to add a Template.
2. Open the Library Browser and activate the Library containing the Templates you want to add.
3. Start the appropriate Add Wizard.
4. Navigate through the available Frameworks in the Wizard and select the Template you want to add. Click *Next* to proceed.
5. Confirm or modify the *Name Property* and *File Name* properties, if desired. Click *Finish* to add the Template.



Form Wizard will create the following Form for you:

Template Name: About Box

Name Property: frmAbout

File Name: about.frm

Click Finish when you are ready to add the Form Template.

Modifying the Name Property, and File Name

Adding Form Templates

The Form Wizard adds Form Templates from your Library into a Visual Basic project.

To add a Form Template to your project using the Form Wizard, proceed as follows:

1. Make sure the project to which you want to add a form is open.
2. Bring up the Form Wizard by selecting *AppFramework + Add + Form* from the Visual Basic *Add-Ins* menu. The *Add Form - Step 1* dialog appears.
3. Locate the Form Template you want to add in the TreeView. This is done by expanding the Frameworks in the TreeView to display the Form Templates contained within. You can also use the *Find* facility to locate a Template.

   Expanding a Form Template in the TreeView lets you view additional information about the Template such as the Comment and Name property. Also, as you select the Template, you can see the author and last date modified in the status bar at the bottom of the dialog.

   Select the desired Form Template and click *Next*. The *Add Form - Step 2* dialog will appear.
4. Change the File Name and/or the Name Property for the Form to be added to the values you want to appear in your project, *or* accept the default values supplied from the Library. If you change the file name, all related FRX file references (if any) will automatically be changed for you.

Click *Finish* to add the Form Template to your active Visual Basic project. An Add Form status dialog is displayed to conform your successful add. Click *OK* to return to your work in Visual Basic.

Adding Class Templates

The Class Wizard is used to add Class Templates to your Visual Basic project. Bring up the Class Wizard by selecting *AppFramework + Add + Class Module*. From this point, the rest of the process is identical to that for adding Form Templates (see previous topic).

Adding Code Templates

The Code Wizard is used to add Code Templates to your Visual Basic. Bring up the Code Wizard by selecting *AppFramework* + *Add* + *Code Module*. From this point, the rest of the process is identical to that for adding Form Templates (see previous topic).

Adding Control, Menu, and Function Templates

This topic describes how to add Control and Menu Templates to forms in a Visual Basic project, and how to add Functions to code windows.

During the capture process, the methods for controls and menus are captured along with the object definition.   When you add Control or Menu Templates back to a project, you have the option of including these methods or not.   You can also insert a comment in the methods to help you identify code that was automatically inserted with the items by AFW.

Another useful tool lets you instantly create copies of controls and methods.   This is the *Clone Controls* tool.   When you clone a collection of controls, the collection does not get saved to a library.   This tool is discussed in a later topic.

Contents

Adding Controls

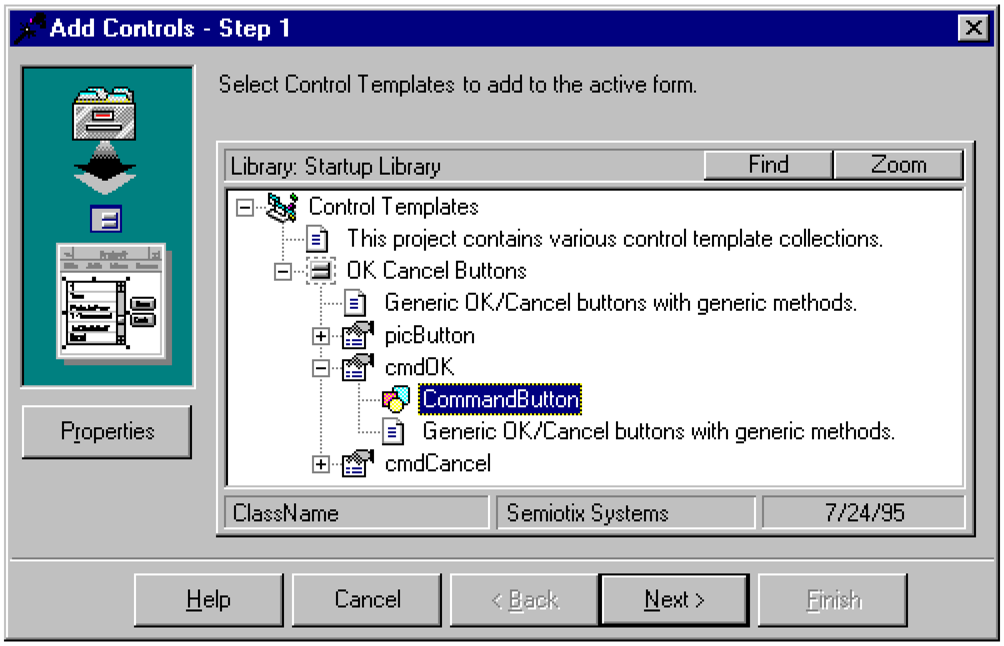
The Control Wizard is used to add a Control Template or a collection of Control Templates to your project.

To add a Control Template or Control Templates Collection to a form in your active project, proceed as follows:

1. Open the form in Visual Basic on which you want to place the Controls Collection. Make sure the form is active and has the input focus.
2. Start the Control Wizard by selecting *AppFramework* + *Add* + *Controls* from the Visual Basic *Add-Ins* menu. The *Add Controls - Step 1* dialog appears.
3. Expand the Frameworks in the TreeView to list the Controls Collections contained within. Expand any Collection to view the individual Control Templates contained within. To add an entire Collection of Controls to your form, select a Collection in the TreeView and click Next. To add an individual Control Template to your form, expand the applicable Collection node in the TreeView, select an individual Control Template and click *Next*. The *Add Controls - Step 2* dialog is displayed.

Click *Finish* to add the control(s). The Add Controls status dialog confirms your successful capture. Click *OK* to return to your work in Visual Basic.

Note You can add individual control templates that are part of a collection. To do so, open the collection and select the desired template. Follow the wizard steps to add the individual control template.



Select Control Templates to add to the active form.

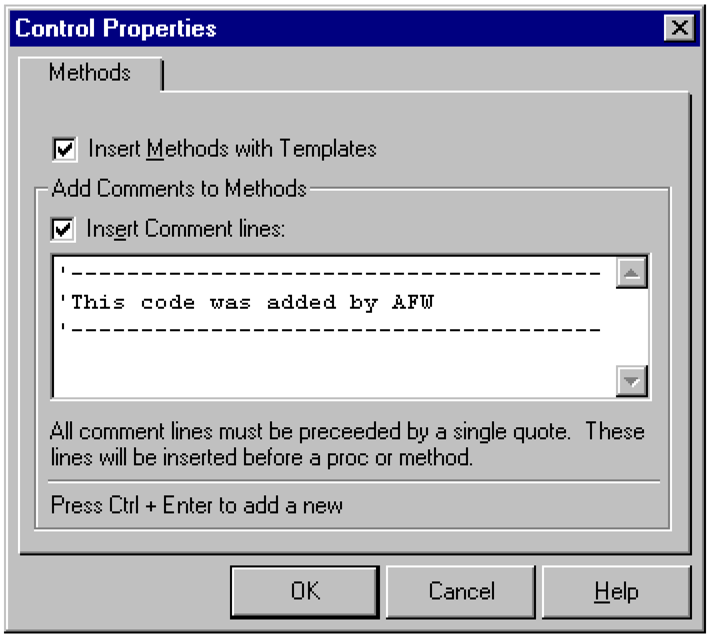
Library: Startup Library

Control Templates
This project contains various control template collections.
OK Cancel Buttons
Generic OK/Cancel buttons with generic methods.
picButton
cmdOK
CommandButton
Generic OK/Cancel buttons with generic methods.
cmdCancel

ClassName | Semiotix Systems | 7/24/95

A Collection of Control Templates

While you are in *Step 1* or *Step 2* the Control Wizard, you can view and modify certain Control Wizard options by clicking on the *Properties* button in either dialog. The Control Properties dialog is displayed. In this dialog, you can specify whether the methods should be included with the Control Templates you are adding. If you include methods, you also elect to have AFW include a set of comment lines with them. These lines will be placed in the code window before each method to help identify code that was inserted by AFW.



[✓] Insert Methods with Templates

Add Comments to Methods

[✓] Insert Comment lines:

```
'---------------------------------------
'This code was added by AFW
'---------------------------------------
```

All comment lines must be preceeded by a single quote. These lines will be inserted before a proc or method.

Press Ctrl + Enter to add a new

Control Wizard Properties

Note When you add a Control Template to a form that requires a VBX/OCX Reference, AFW automatically adds the Custom Control Reference to your project before it adds the control. *Also:* the property TabIndex cannot be set properly due to the fact that the order controls are added may result in erroneous settings of this property.

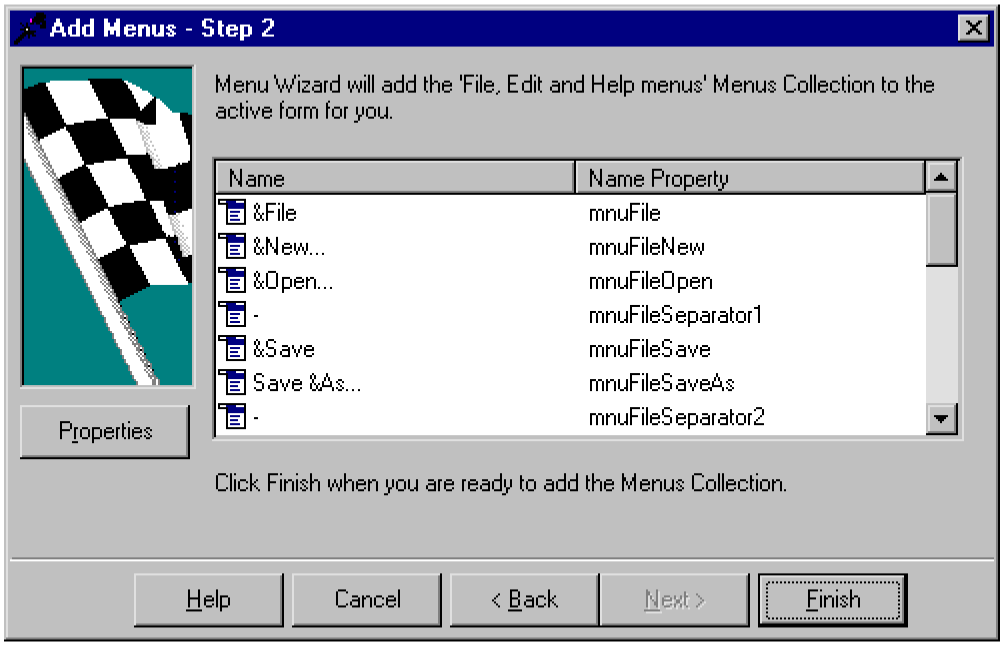
Adding Menus

The Menu Wizard is used to add a Menu Template or a collection of Menu Templates to your project.

To add menus to a form using the Menu Wizard:

1. Open the form in Visual Basic on which you want to place the menus. Make sure the form is active, and has the input focus.
2. Start the Menu Wizard by selecting *AppFramework* + *Add* + *Menus* from the Visual Basic *Add-Ins* menu. The *Add Menus - Step 1* dialog is displayed.
3. Expand any Framework in the TreeView to view the Menu Collections contained within. Open any collection to view the individual Menu Templates contained within. Select the Menus Collection or individual Menu Template you would like to add and click *Next*. The *Add Menus - Step 2* dialog is displayed.
4. Click *Finish* to add the menu(s).

The Menu Wizard offers the same options as the Control Wizardnamely, to add methods with your menus or not, and to add comment lines with your methods or not. As in the Control Wizard, these options can be modified by clicking the Properties button on either the *Step 1* or *Step 2* dialog.



Menu Wizard will add the 'File, Edit and Help menus' Menus Collection to the active form for you.

Name	Name Property
&File	mnuFile
&New...	mnuFileNew
&Open...	mnuFileOpen
-	mnuFileSeparator1
&Save	mnuFileSave
Save &As...	mnuFileSaveAs
-	mnuFileSeparator2

Click Finish when you are ready to add the Menus Collection.

Menu Wizard Summary

Adding Functions

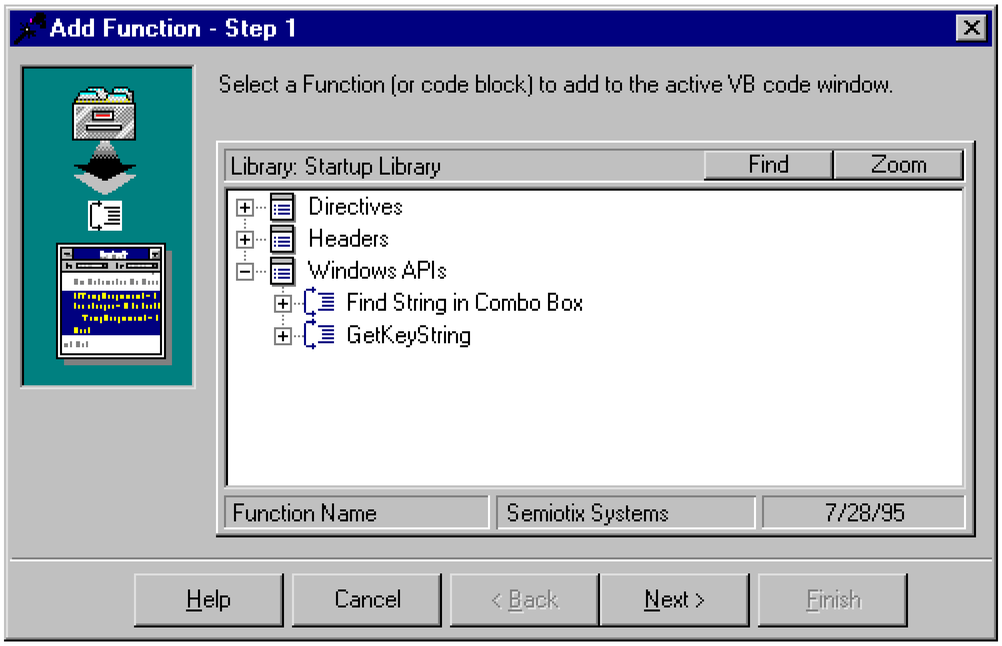
The Function Wizard is used to add a Function to your project.

Remember, Functions in AFW can be comprised of complete procedures or any other contiguous code block. For example, a standard comment header that you want to appear in every module can be captured as a Function.

To add a Function to your project using the Function Wizard:

1. Select *AppFramework* + *Add* + *Function* from the Visual Basic *Add-Ins* menu. The *Add Function - Step 1* dialog is displayed.
2. The information in the TreeView control on this dialog is organized differently than in the previous Wizards. AFW Function Templates are global to a Library and are NOT associated with specific Frameworks. Consequently, there are no Frameworks displayed in the TreeView. Rather, Functions are grouped into Categories. These Categories make up the first level of the TreeView.
3. Expand the Categories to view the Functions contained within. Select the Function you want to add and click *Next*. The *Add Function - Step 2* dialog is displayed.
4. The *Step 2* dialog lets you view your Function before applying it and even edit it, if you want. When you are ready, click *Copy to Clipboard* to paste the Function to the Windows clipboard.

Click *Finish* to end the Wizard. At this point, you can set the insertion point at any position in the code window and paste your Function into the window by selecting *Edit* + *Paste* from the Visual Basic menu, *Ctl* + *V* from the keyboard, or right mouse clicking.**Tip** The zoom is especially useful when viewing functions. Highlight the tree item entitled [Code Sample] and click *Zoom*. The entire function can now be viewed.



Select a Function (or code block) to add to the active VB code window.

Library: Startup Library

- Directives
- Headers
- Windows APIs
  - Find String in Combo Box
  - GetKeyString

Function Name | Semiotix Systems | 7/28/95

Step 1 of the Function Wizard

Just as with Control and menu Templates, you can have AFW add a set of comment lines to each Function by setting the Function Wizard *Properties*. This option is available by clicking on the *Properties* button in the *Step 2* dialog. Comments are useful in helping you identify code that has been added from the Library.

Applying Styles

This topic describes the process of applying Styles to Visual Basic objects using the Style Wizard.

Another tool is provided that allows you to apply styles instantly. This is the *Clone Styles* tool. When you clone a style, the style is not saved to a library. This tool is discussed in a following topic.

Contents

Overview of Applying Styles

During the capture process, you captured the Style from a single form or control that was selected. When applying Styles however, you can apply them to multiple controls at the same time. Of course, since only one form can be selected at a time, form Styles can only be applied to one form at a time.

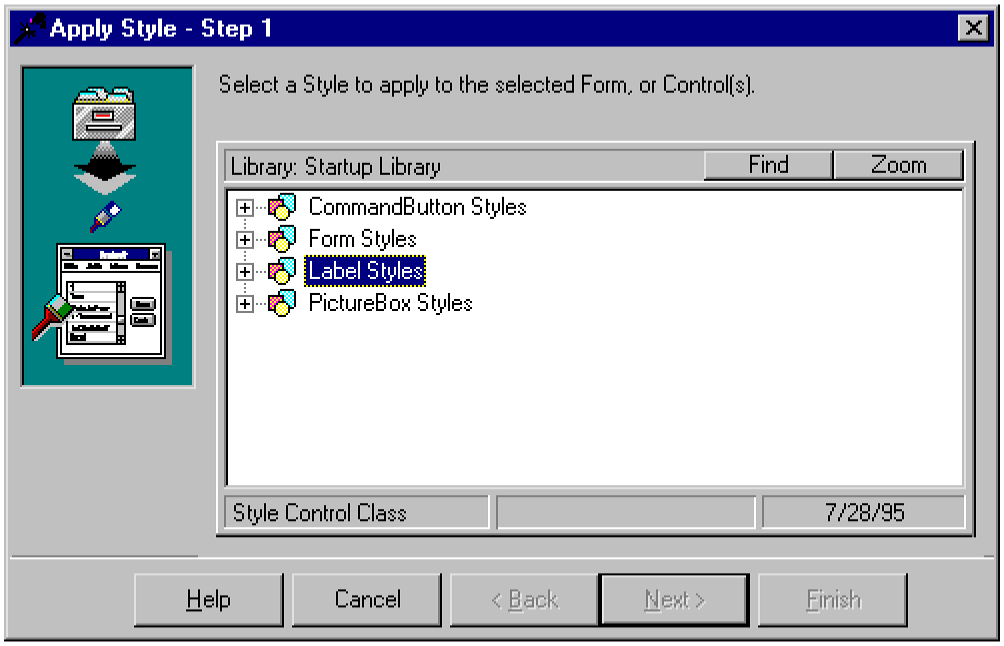
As with the other Wizards, you must first select the object(s) you will be working with in Visual Basic before applying a Style. Remember: Styles are collections of property settings for a particular object class (that is, a form or a particular type of control). Therefore, they can only be applied to objects of the same class. If you select a group of objects to which you want to apply a Style and your selection includes objects of the wrong class, the Wizard will automatically ignore those objects in applying the Style.

Tip When selecting forms, click on the form itself. Make sure the class name Form appears in the drop-down in the properties window of Visual Basic.

To apply a Style to one or more Visual Basic objects using the Style Wizard:

1. Select the object(s) to which you want to apply the Style.
2. Start the Style Wizard by selecting *AppFramework + Apply Styles* from the Visual Basic *Add-Ins* menu. The *Apply Style - Step 1* dialog is displayed.
3. Like Functions, AFW Styles are global to a Library; they are NOT associated with a specific Framework. Rather, Styles are grouped into *object classes*. Since you apply Styles to objects of a particular class, this organization helps you easily locate the correct Style to apply.
4. Expand any object class in the TreeView to view all the Styles contained within. Select the desired Style and click *Next*. The Apply Style - Step 2 dialog is displayed.
5. The Step 2 dialog displays a list of the properties included in the selected Style. If the list does not look like the Style you want to use, click *Back* and select a different Style. If satisfied with your selection, click *Finish* to apply the Style. An Apply Style status dialog is displayed to confirm the successful application of your Style.

Note As you are applying a Style, you may receive an error message if there is an object class mismatch between the Style and one or more of the selected objects. In such cases, AFW WILL apply the Style to all objects of the appropriate class and skip those of the wrong the class.



Select a Style to apply to the selected Form, or Control(s).

Library: Startup Library

- CommandButton Styles
- Form Styles
- Label Styles
- PictureBox Styles

Style Control Class | | 7/28/95

A List of Available Style Control Classes

Adding an Application Framework

A *Project Framework* is a collection of Templates and associated references and files that can be added to a Visual Basic project at the same time. Project Frameworks can be used to create starting frameworks for applications, such as the MDI and SDI frameworks that ship with the AFW *Professional Frameworks* Library. They can also be used to group logically related collections of components, like a collection of forms, that typically work together in an application. Finally, they can be used to create a particular programming environment by adding the desired custom controls, references, and basic forms and modules. In the latter case, a Framework works much like the Autoload projects in Visual Basic. Unlike Autoload, however, you can create as many Frameworks as you want in AFW.

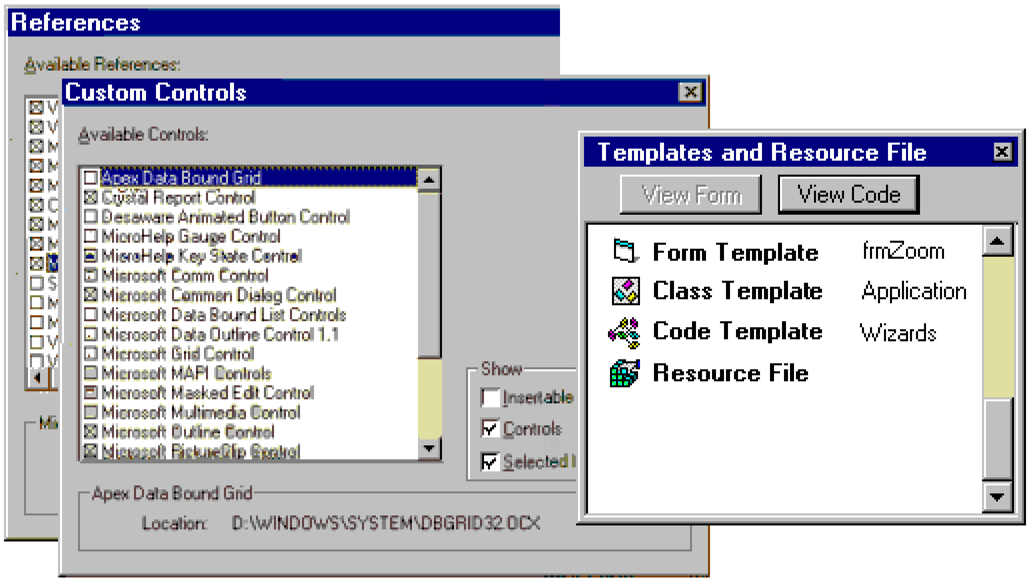
Tip If you plan to create and use Frameworks as startup programming environments for Visual Basic projects, create a Visual Basic Autoload project that is devoid of custom controls, references and modules (Visual Basic always places a startup form in the project.) Then add a Framework to have AFW create a starting environment for you.

Contents

Framework Items

An *Application Framework* consists of the following items that are associated with a particular *Project Template*:

- Form Templates
- Class Templates
- Code Templates
- Associating Custom Controls
- Associated References
- Associated Resource File
- Associated Files



☐ Apex Data Bound Grid
☒ Crystal Report Control
☐ Desaware Animated Button Control
☐ MicroHelp Gauge Control
☒ MicroHelp Key State Control
☐ Microsoft Comm Control
☒ Microsoft Common Dialog Control
☐ Microsoft Data Bound List Controls
☐ Microsoft Data Outline Control 1.1
☐ Microsoft Grid Control
☐ Microsoft MAPI Controls
☒ Microsoft Masked Edit Control
☐ Microsoft Multimedia Control
☒ Microsoft Outline Control
☒ Microsoft PictureClip Control

☐ Insertable
☑ Controls
☑ Selected

Apex Data Bound Grid
Location: D:\WINDOWS\SYSTEM\DBGRID32.OCX

Form Template	frmZoom
Class Template	Application
Code Template	Wizards
Resource File	

Framework Items (except files)

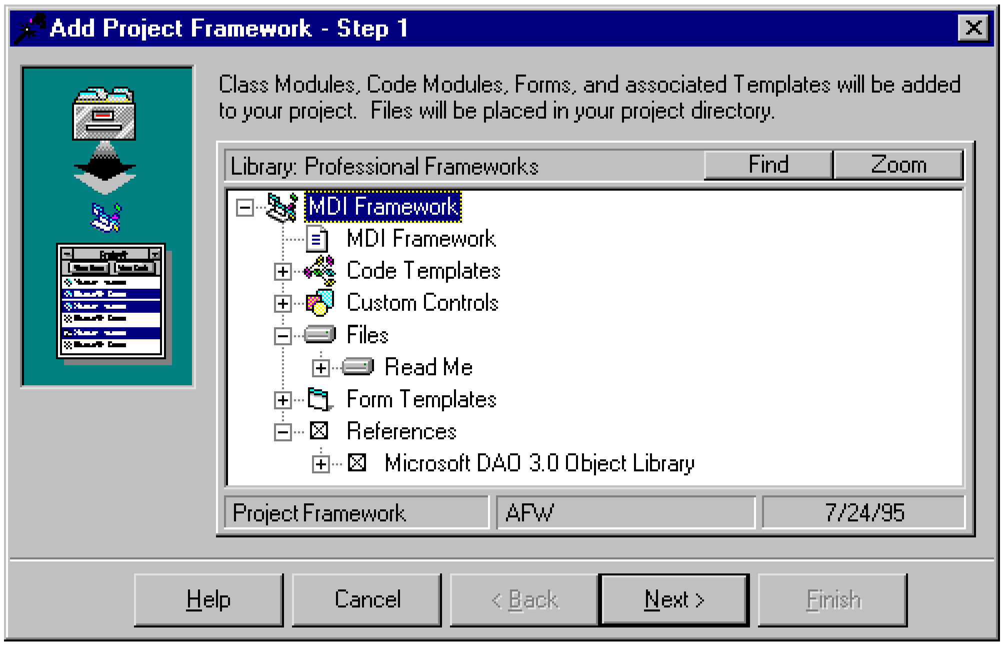
As discussed earlier, Functions and Styles are global to a Library and are not part of a particular *Project Framework*.

Although AFW organizes Control and Menu Templates under *Project Frameworks*, these objects are NOT added to your Visual Basic projects when you lay out Frameworks. This is because controls and menus must be contained on a form. There is no way for AFW to know which form they should be added to when the Framework is laid out.   Including Control and Menu Templates within Project Frameworks is useful for organizing your Templates, but you must add them to projects using the Add Controls or Add Menu Wizards.

Tip   AFW seeds each Library you create with a Framework called Public Project Framework.   This is the ideal place to store Control and Menu Templates that you plan to use globally across many projects.

Adding a Framework

Adding an entire Framework to a Visual Basic project is just as easy as adding an individual Template or applying a Style. Before you add your Framework, make sure you have opened the Visual Basic project in which you want to place all of the Templates, Custom Controls, References, and Resource Files contained within it. If you are working with a new project, you must save it first. This is required so AFW knows where to place the applicable files.



Class Modules, Code Modules, Forms, and associated Templates will be added to your project. Files will be placed in your project directory.

Library: Professional Frameworks

```
MDI Framework
    MDI Framework
    Code Templates
    Custom Controls
    Files
        Read Me
    Form Templates
    References
        Microsoft DAO 3.0 Object Library
```

Project Framework | AFW | 7/24/95

An Application Framework

To add a Framework to a Visual Basic project:

1. With your project open and saved, select *AppFramework + Add + Framework* from the *Add-Ins* menu in Visual Basic. The *Add Project Framework - Step 1* dialog is displayed.
2. The TreeView displays all available Frameworks within the Active Library. As you expand each Framework, you can review the individual Templates and Associations contained within it. You must, however, add the entire Framework with this Wizard. If you want to add individual Templates, you must use the corresponding Add Wizard to do so.
3. Select the Framework you want to use and click *Next*. The *Add Project Framework - Step 2* dialog is displayed. This dialog provides guidance on what to expect in the proceeding steps.
4. Click *Next*. The *Add project Framework - Step 3* dialog will be displayed. You have one last chance to review the Framework contents here and either cancel or proceed.
5. Click *Finish* to add the Framework. A Status dialog will be displayed that itemizes the successful addition of each object in the Framework or any errors encountered. An error condition for any object will cause the add for that object (and only that object) to be skipped. In other words, AFW will add everything from the Framework it can.

Note You must have the applicable custom controls or OLE automation objects installed and registered on your system before you can add their associated references. AFW doesnt actually store or copy these objects, it merely associates their IDs with your project.

AFW adds your Framework contents in the following order:

1. All Custom Controls and References will be added to the project.
2. All Forms, Class Modules, and Code Modules will be added to the project.
3. The Resource File will be added to the project.
4. All External Files will be placed in the active project directory.

Of course, some of these steps will be skipped if the Framework you are adding does not contain the corresponding items.

Adding a Framework is as easy as that!   This capability can be used to give any new application a real head start. Creating and using Frameworks will pay for your investment in AFW hundreds of time over. Remember, several time saving Frameworks from Semiotix Systems are provided in the Libraries that ship with AFW.   These are designed to be added to projects immediately and run.

Tip   Always provide a Sub Main() when creating a framework.   In this way you can immediately run the project after the framework has been added.

Cloning Controls, Forms, and Styles

This topic discusses the use of AFWs cloning tools. These tools allow you to create copies of forms and controls, *including their methods*, and copies of Styles without first saving them in an AFW Library.

Why would you want to do this? There may be many times as you develop applications when you need to replicate objects or Styles within your current project, but arent sure about adding them to a Library. Perhaps you arent sure they have longer term reusability value or your organization requires them to be reviewed and approved before adding them to a shared Library. Cloning is the answer in these cases. Cloning gives you virtually all the same features as capturing and applying Styles and Templates, but without the Library storage process in between.

The Cloning tools are stand-alone tools. They operate independent of your Libraries with one exception: when you clone a Style, the Default Style for that object class (stored in the Active Library) dictates which properties are cloned.

Contents

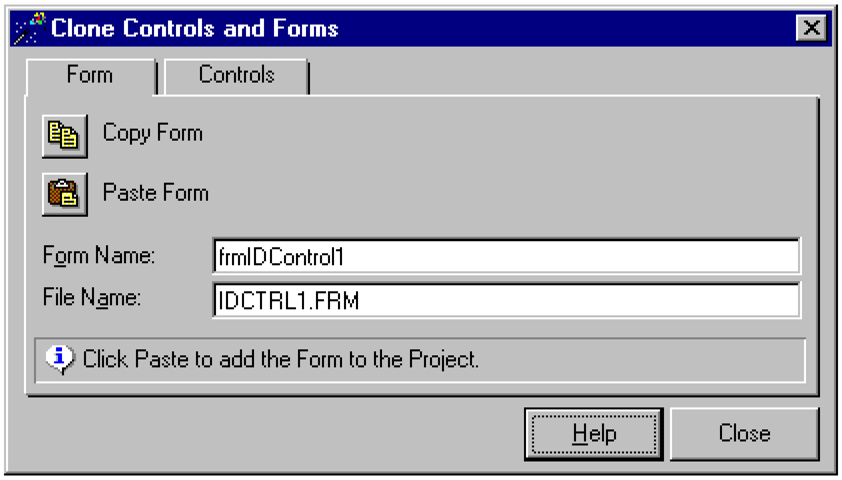
Cloning Forms

Have you tried to duplicate a form in Visual Basic? This usually involves designing the form the way you like it, doing a *File* + *Save As* in Visual Basic, quitting the project, editing the form file in a text editor, and so on. Basically, its not an easy task. However, the AFW Cloning tool can quickly make a duplicate of any form. Cloning a form is much like a copy-and-paste operation.

To clone a form:

1. Make sure the form you want to clone is open in your Visual Basic environment and has the input focus. If youve made any changes to the form, be sure to save it before starting the cloning operation.
2. Select *AppFramework* + *Clone* + *Controls and Forms* from the *Add-Ins* menu. The Clone Controls and Forms dialog is displayed.
3. Make sure that you have tabbed to the Form page of the dialog. Click *Copy Form*. If the selected form is dirty, youll get a message asking whether you want to proceed with the capture by saving the prior version of the form (if it exists) or canceling the operation.
4. You can now provide a new name for the form and its associated file in Visual Basic. Defaults for these names are provided by AFW. Accept these defaults or provide names of your own. Remember that the form and file names must not conflict with names of existing forms in your Visual Basic project or other file names on your disk.

Click the *Paste Form* button to complete the operation. An exact copy of the form will be added to your Visual Basic project with the names you specified. If the form had an associated FRX file, that file will also be cloned and all references between it and the FRM file updated.



Form Name: frmIDControl1
File Name: IDCTRL1.FRM
Click Paste to add the Form to the Project.

Cloning a Form

Note The clone operation uses the physical file associated with the form. For this reason you should save the form file before you clone it.

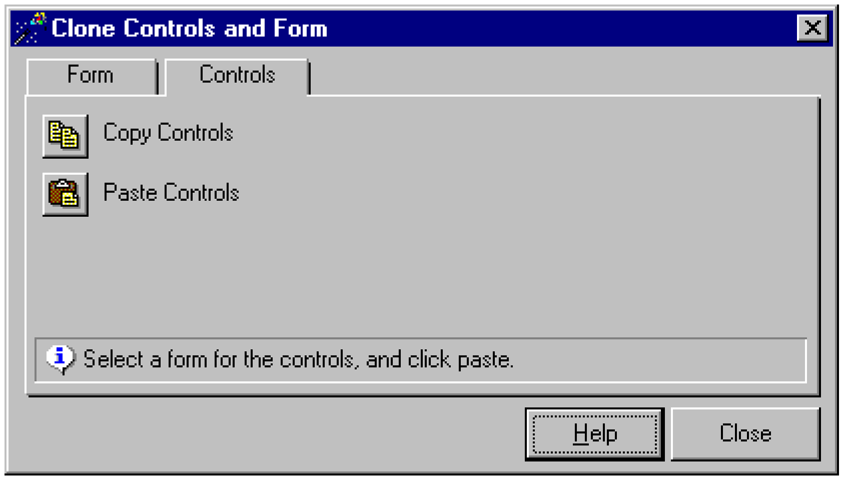
Cloning Controls

You might wonder why you should clone controls when Visual Basic has a perfectly good mechanism to copy-and-paste controls. The answer is simple: when you copy-and-paste controls in Visual Basic, you lose their methods! When you clone controls using AFW, all methods associated with the controls ARE captured and copied to the new controls.

Cloning controls is similar to cloning forms:

1. Open a form in Visual Basic and select the controls you want to clone.
2. Select *AppFramework + Clone + Controls and Forms* from the *Add-Ins* menu. The *Clone Controls and Forms* dialog is displayed.
3. Click on the *Copy Controls* button on the dialog. When the capture part of the cloning process is completed, a message informs you that your controls are ready to be pasted onto another form and the *Paste Controls* button will be enabled.

Select the form on which you want the controls to be placed, then return to the cloning dialog and click *Paste*. An exact copy of all of the controls and their associated methods will be placed on that form.



Select a form for the controls, and click paste.

Cloning Controls

Note Due to a limitation in the Visual Basic IDE, methods must be extracted from the physical file associated with the form containing the controls you are cloning. For this reason you should save the form file before you clone them.

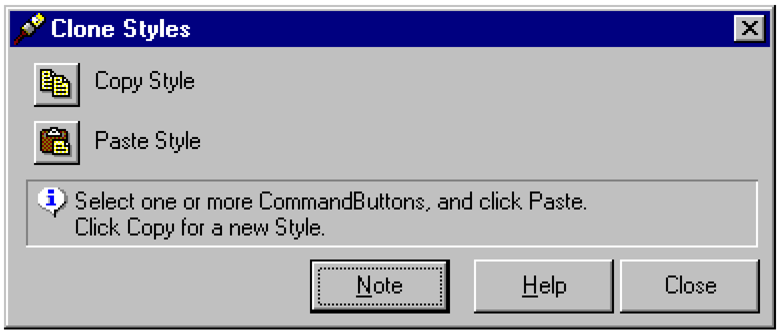
Cloning Styles

The Clone Styles tool allows you to instantly copy and paste Styles between forms or controls.

To clone a Style:

1. Select the form or single control in Visual Basic that has the Style you want to clone.
2. Select the *AppFramework* + *Clone* + *Styles* from the *Add-Ins* menu. The *Clone Styles* dialog is displayed.
3. Click on the *Copy Style* button. When the capture part of the cloning process is completed, a message informs you that the Style is ready to be applied and the *Paste Style* button will be enabled.
4. If you copied properties from a form, select a form then return to the *Clone Styles* dialog and click *Paste Style*. If you copied properties for a control, select one or more controls of the same class, then return to the *Clone Styles* dialog and click *Paste Style*. The following figure shows the tool ready to paste a command button Style.

Tip Keep the Clone Styles tool active to quickly set properties throughout your project. Also use it when you are going through your project to clean up forms or controls. Once you copy a Style, you can paste it as many times as you want. If you want to clone a new Style, select the Visual Basic object and click *Copy* again.



Select one or more CommandButtons, and click Paste.
Click Copy for a new Style.

Ready to Paste a Cloned Style

Note When you clone a Style, AFW copies settings for the properties defined in the *Default Style* for the control class selected. To change the Default Style, use the Library Browser.

Using Find, and Zoom

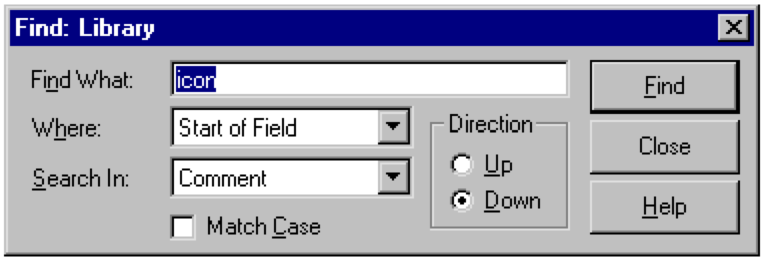
This topic discusses the *Filter*, *Find*, and *Zoom* utilities that are found throughout VB AppFramework. These utilities work in conjunction with the TreeView control in the Wizards to help you manage the Templates and Styles you will be using.   The Library Browser and all Wizards have the *Find* and *Zoom* buttons above the TreeView.

Contents

Finding Templates

This utility lets you find a particular Template or Style in your Wizard lists and in the Library Browser. The *Find* form is show in the figure below. Your choices for finding Templates are saved, and these choices will be shown in the form the next time you use it.

Finding an item is simple. Type a few characters in the *Find What:* field and Click the *Find* button. In this example, you would be looking for those characters in the *Start of Field* and the find utility will be searching <All Fields> from your current location down. You can select various options in the *Where:* and *Search In:* combo boxes to further narrow down your search. Dont worry if the TreeView is currently collapsed. If an item you are looking for is found, the tree will expand and the found item will be selected.


Find: Library

Find What: icon

Where: Start of Field

Search In: Comment

Match Case [ ]

Direction: Up [✓ Down]

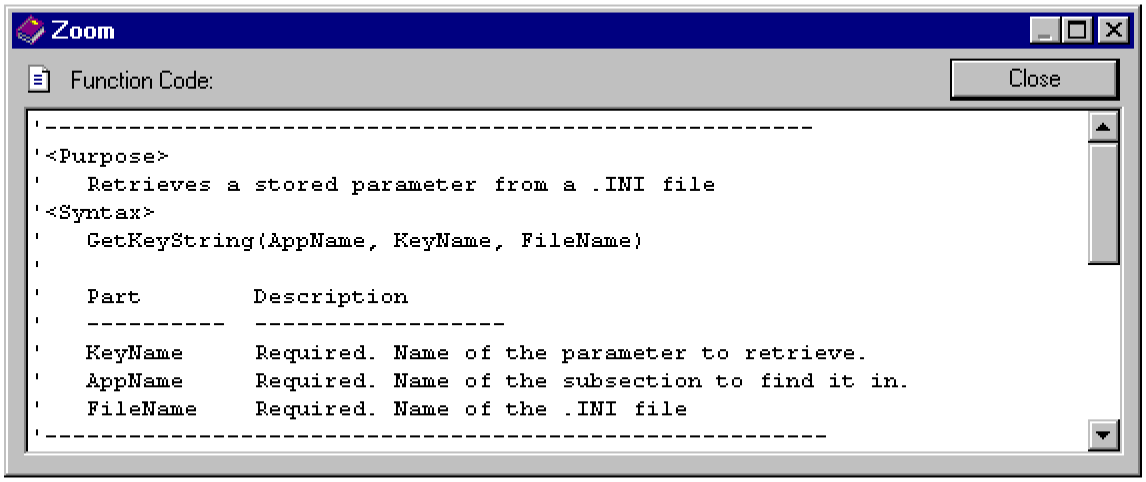
The Find Form

Zooming on Data

The zoom utility lets you view the tree data in a form. This is useful to see all of the information contained in long fields, such as comments and code. There are two ways you can zoom on an item.

1. Click the *Zoom* button that is found on the top right of the trees.
2. Click your right mouse button while your cursor is over the tree and select *Zoom* from the pop-up menu.

The Zoom form can be sized or maximized to view all of the data. The following figure shows the form when zooming on a Functions code.



```
'----------------------------------------------------------
'<Purpose>
'   Retrieves a stored parameter from a .INI file
'<Syntax>
'   GetKeyString(AppName, KeyName, FileName)
'
'   Part        Description
'   ----------  ------------------
'   KeyName     Required. Name of the parameter to retrieve.
'   AppName     Required. Name of the subsection to find it in.
'   FileName    Required. Name of the .INI file
'-----------------------------------------------------------
```

Zooming on a Function

In some cases the item that you are zooming on is not an actual Template, but a folder that contains Templates. This is true when zooming on Style control classes or a Project Framework. In these cases, the zoom form will include a list of Templates contained within that folder.

Visual Basic Integrated Development Environment. The development environment provided by Visual Basic that allows Add-Ins to manipulate Visual Basic objects at design time.

A database that contains organized collections of templates and styles; Physically a Microsoft Access MDB.

Optional. Typically the name of the individual that is creating the item, however this field can be used for any arbitrary purpose.

A Template that is used to group related Templates and Associated items.   All items can be applied at the same time using the Framework Wizard.   Similar to a Visual Basic project.

A group of related items, such as controls, or menus. Oftentimes there is a hierarchy of items within the collection.

The Library that has been so selected in the browser for use.   Of all of the available libraries, only one can be active at any time.

A default group of properties for a particular class of control. This is defined for each library in the Library Browser. This is the set of properties that is by default used when capturing and cloning styles. You can create a Custom Style during the capture process.

A distinct kind of control that is used within Visual Basic. Each control class appears in the toolbox as a separate item, and can be seen in the Properties window.

The process whereby an item is assigned to a particular project template. The purpose of this is to build an application framework that can be added using the Frame Wizard.

A shell for an application, or project. Frameworks are constructed to provide accelerated starting points for application development. They can simply be as set of references, and controls, or they can be entire collections of forms, functions, and related files.