Your component will be easier to use if your object model is similar in style to those of other components, and if the methods and properties of your objects have the same names as methods and properties that provide the same functionality for objects provided by other components.

This appendix includes the following topics relating to these issues.

Contents

1

For More Information "Programming with Objects" and "Programming ActiveX Components" contains information you may find helpful in understanding these issues.

Object Naming Guidelines

When selecting names for objects, properties, methods, and events, choose names that can be easily understood by the users of your component. These elements comprise the programming interface of your component — the more clear you make their names, the more usable your code will be.

The rules in this topic apply to names for:

- Objects.
- The properties, methods, and events that comprise the interfaces of your objects.
- Named arguments of properties, methods, and events.

1

Use Entire Words or Syllables Whenever Possible

It is easier for users to remember complete words than to remember whether you abbreviated Window as Wind, Wn, or Wnd. The following table lists two examples of recommended naming conventions.

Use	Don't use
Application	App
SpellCheck	SpChk

2

When you need to abbreviate because an identifier would be too long, try to use complete initial syllables. For example, use AltExpEval instead of either AlternateExpressionEvaluation or AltExpnEvln.

Use Mixed Case

All identifiers should use mixed case, rather than underscores, to separate the words in the identifier. The following table lists two examples of recommended naming conventions.

Use	Don't use
ShortcutMenus	Shortcut_Menus, Shortcutmenus, SHORTCUTMENUS, SHORTCUT_MENUS
BasedOn	basedOn

3

Use Consistent Terminology

Use the same word you use in the interface; don't use identifier names like HWND, which are based on Hungarian notation. Remember that this code will be accessed by other users, so try to use the same word your users would use to describe a concept.

Use the Correct Plural for Collection Class Names

Using plurals rather than inventing new names for collections reduces the number of items a user must remember. It also simplifies the selection of names for collections. The following table lists some examples of collection class names.

Use	Don't use
Axes	Axiss
SeriesCollection	CollectionSeries
Windows	ColWindow

4

For example, if you have a class named Axis, a collection of Axis objects is stored in an Axes class. Similarly, a collection of Vertex objects is stored in a Vertices class. In rare cases where the same spelling is used for both singular and plural, append the word "Collection" — for example, SeriesCollection.

Note This naming convention may not be appropriate for some collections, especially where a set of objects exists independently of the collection. For example, a Mail program might have a Name object that exists in multiple collections: ToList, CcList, and so forth. In this case, you might specify the individual name collections as ToNames and CcNames.

5

Use a Prefix for Your Constants

Select a three- or four-letter, lowercase prefix that identifies your component, and use it on the names of constants your component provides in its type library, as well as on the names of the Enums that define those constants.

For example, a code component that provides loan evaluations might use ‘levs’ as its prefix. The following Enum for loan types uses this prefix. (In addition, the constants include the upper-case characters ‘LT’ to indicate the enumeration they belong to.)

```
Public Enum LoanType
    levsLTMortgage = 1
    levsLTCommercial
    levsLTConsumer
End Enum
```

6

Using a prefix reduces the chance that the constants for your component will have name conflicts with constants for other components. Name conflicts of this type can cause difficult bugs for your users.

The shorter the constant name, the more important this rule becomes. In the worst case — constant names that are common words, like the names of colors — such conflicts become almost inevitable.

Verb/Object vs. Object/Verb

If you create method names that combine a verb with the name of the object it acts on, you should be consistent about the order. Either place the verb before the object in all cases, as with InsertWidget and InsertSprocket, or always place the object first, as with WidgetInsert and SprocketInsert.

Both schemes have their advantages. Verb/object order creates names that are more like normal speech, and thus show the intent of the method better. Object/verb order groups together all the methods that affect a particular object.

It doesn’t matter which order you choose, but mixing the two orders will confuse the users of your component.

Filename: in.doc Project: A Traditional User's Guide Template
Template: Author: Ruth Silverio Last Saved By: Christine
Revision #: 4 Page: 3 of 3 Printed: 09/20/96 07:02 PM