

# Contents

Related



[Language Reference](#)



[Introducing ABC OLE Automation](#)



[Visual Basic Concepts](#)



[Writing a Program](#)



[Running and Setting Up ABC](#)



[Handling ABC Events](#)



[Working with Chart Files](#)



[Working with Objects](#)



[Working with Shapes](#)



[Working with Lines](#)



[Working with Text](#)



[Working with Data Fields](#)



[Using Color](#)



[ABC Object Hierarchy](#)



[Properties and Methods by Task](#)



[ABC Features Not Automated](#)



[Using C++ with ABC OLE Automation](#)

## Related Topics

### Description

[ABC FlowCharter Help Contents](#)

[Conventions](#)

[Searching in Help](#)

[Using the Help System](#)

[Visual Basic Help](#)

[Other Programs in ABC FlowCharter 4.0](#)

*For Help on Help, Press F1.*

## Other Programs in ABC FlowCharter 4.0

Related

ABC OLE Automation is one of the programs in the ABC FlowCharter package.

**Click a button to go to the help system for the program you want to learn about.**



ABC FlowCharter 4.0 is the latest version of the popular charting program. It provides the tools you need for Total Process Management and for all your other charting needs.



ABC DataAnalyzer 1.1 give you the tools to draw data-driven charts, including histograms, run charts, Pareto charts, control charts (including P charts, PN charts, C charts, U charts, Moving Range, and X Avg, R charts), scatter charts, and pie charts.



ABC SnapGraphics 2.0 is the quick tool for creating graphics for all your business needs.



ABC Viewer 1.0 is the latest program in the ABC family. It lets anyone view and work with the charts created using ABC FlowCharter.



ABC OLE Automation is an advanced feature of ABC FlowCharter 4.0 that requires Visual Basic, knowledge of programming in Visual Basic or C++, and knowledge of ABC FlowCharter. You can control ABC FlowCharter and ABC Viewer using OLE Automation.



If you have the CD ROM version of ABC FlowCharter 4.0, you will find copies of Designer and Picture Publisher included on it. These special versions have the full functions of the retail versions. After you have used them for 30 days, however, they will stop working. Contact Micrografx or your dealer to purchase retail versions. See the [Read Me](#) for more information.

## Related Topics

Description

[Contents](#)



## Introducing ABC OLE Automation

### Related

Welcome to ABC OLE Automation!

ABC OLE Automation™ is a powerful tool you can use to customize ABC FlowCharter™ and ABC Viewer to meet your own specific needs. The extensive power and flexibility of Automation give you endless control over ABC FlowCharter and ABC Viewer.

ABC OLE Automation can provide seamless integration with outside applications. You can write automation programs that use ABC information to perform tasks in other applications or use data from other applications to create and manipulate ABC charts.

### [Running and Viewing the Sample Files](#)

[How to Use this Help](#)

[Using the Help System](#)

[Searching in Help](#)

[Jumping to Visual Basic Help](#)

[Controlling ABC FlowCharter and ABC Viewer](#)

[Conventions](#)

## Related Topics

Description

[Contents](#)

## Using the Help System

Related

This help system is contained in the file AUTOMATE.HLP. If you wish, you can order the ABC OLE Automation Programmer's Guide with the order form included in your copy of ABC FlowCharter. Alternatively, you can order the book by calling 1-800-603-0074. The guide is a printed version of the information included in the help system.

There are several ways you can access the ABC OLE Automation help file. You can run the file from the help that appears when you access help from ABC FlowCharter. You can run it using the Windows File Manager. You can run it using Excel 5.0 (which also lets you browse through the language elements and see quick descriptions of them).

### **Click a topic to learn how to access ABC OLE Automation help**

[Accessing ABC OLE Automation Help from ABC FlowCharter Help](#)

[Accessing ABC OLE Automation Help from Visual Basic](#)

[Accessing ABC OLE Automation Help using Excel 5.0](#)

[Accessing ABC OLE Automation Help using the File Manager](#)

## Related Topics

### Description

[Introducing ABC OLE Automation](#)

[Jumping to Visual Basic Help](#)

[Searching in Help](#)

## Accessing ABC OLE Automation Help from ABC FlowCharter Help

### Related

### Automation

This help for ABC OLE Automation is linked to the ABC FlowCharter help. To the right of selected buttons, commands, dialog boxes, and areas of dialog boxes are buttons labeled "Automation." If you click one of those buttons, you go to the ABC OLE Automation help for a related property or method. Be sure to check the Related Topics in the ABC OLE Automation help to find other properties or methods that are related to the ABC FlowCharter area you are working in.

### ABC Equivalent

When a property or method has an equivalent ABC FlowCharter command, a button labeled "ABC Equivalent" appears to the right of the topic title. You can click on the button to go to the ABC FlowCharter topic that includes a description of the related command, button, option, or other ABC FlowCharter element.

### ABC Equivalent

If no button or a gray button appears to the right of a topic, then there is not an ABC FlowCharter equivalent.

### To access the ABC OLE Automation help file from ABC FlowCharter help:

1. Click the **Automation** button immediately to the right of the command, option, or area that you want to know the ABC OLE Automation equivalent for. For example, if you want to know the equivalent for the Align to Rulers option in the Preferences dialog box, go to the topic Alignment Options. Find the paragraph about Align to Rulers and click the **Automation** button. The ABC OLE Automation help file runs showing the topic on the equivalent ABC OLE Automation property or method.
2. To return to the ABC FlowCharter help from the ABC OLE Automation help, click on the button **ABC Equivalent** that appears at the right of the topic title.

## Related Topics

### Description

[Accessing ABC OLE Automation Help from Visual Basic](#)

[Accessing ABC OLE Automation Help using Excel 5.0](#)

[Accessing ABC OLE Automation Help using the File Manager](#)

[Controlling ABC FlowCharter and ABC Viewer](#)

[Jumping to Visual Basic Help](#)

[Searching in Help](#)

[Using the Help System](#)

## Accessing ABC OLE Automation Help Using the File Manager

Related

To access ABC OLE Automation help using the File Manager:

1. Open the Windows File Manager.
2. Locate the directory where ABC FlowCharter is installed (C:\ABC, by default).
3. Double click the file AUTOMATE.HLP. The ABC OLE Automation help file runs showing the Contents topic.

## Related Topics

### Description

[Accessing ABC OLE Automation Help from Visual Basic](#)

[Accessing ABC OLE Automation Help using Excel 5.0](#)

[Accessing ABC OLE Automation Help using the File Manager](#)

[Jumping to Visual Basic Help](#)

[Searching in Help](#)

[Using the Help System](#)

## Accessing ABC OLE Automation Help Using Excel 5.0

Related

To access ABC OLE Automation help using Excel 5.0:

1. Run Excel 5.0.
2. Open the Insert menu and choose the Macro command. A submenu opens.
3. Choose Module. The module interface displays.
4. Open the Tools menu and choose the References command. The References dialog box opens.
5. Click ABC OLE Automation 1.0 so an X appears in the box in front of it.
6. Click OK to close the dialog box.
7. Open the View menu and choose the Object Browser command. The Object Browser dialog box opens.
8. Click the down arrow to the right of the Libraries/Workbooks text box. A list of available libraries and workbooks appears.
9. Click ABC FlowCharter to select it. The Objects/Modules list box shows the objects available in ABC FlowCharter OLE Automation.
10. Click the object you want information about in the Objects/Modules list box. A short explanation appears at the bottom of the dialog box.
11. Click the button with a question mark in it, if you wish. ABC OLE Automation help appears showing the topic about that object.  
*or*  
Click the method or property you want information about in the Methods/Properties list box. A short explanation appears at the bottom of the dialog box.
12. Click the button with a question mark in it. ABC OLE Automation help appears showing the topic about that method or property.

**Note:** If the button with a question mark in it is gray, then the help file is not installed. You must install the help file using the Installation program.

## Related Topics

### Description

[Accessing ABC OLE Automation Help from ABC FlowCharter Help](#)

[Accessing ABC OLE Automation Help from Visual Basic](#)

[Accessing ABC OLE Automation Help using the File Manager](#)

[Jumping to Visual Basic Help](#)

[Searching in Help](#)

[Using the Help System](#)

## Jumping to Visual Basic Help

Related

You can jump to the Visual Basic help, at the Contents topic, by clicking [Visual Basic Help](#). However, the jump will succeed only if you have the Visual Basic directory in your DOS path, or if you copy the Visual Basic help file (VB.HLP) to your ABC FlowCharter directory.

## Related Topics

[Accessing ABC OLE Automation Help from ABC FlowCharter Help](#)

[Accessing ABC OLE Automation Help from Visual Basic](#)

[Accessing ABC OLE Automation Help using Excel 5.0](#)

[Accessing ABC OLE Automation Help using the File Manager](#)

[Introducing ABC OLE Automation](#)

[Using the Help System](#)

## Searching in Help

Related

You can use the help  button to find information about the properties, methods, and events in ABC OLE Automation. Each language element has up to two listings: one for a description of the element and one for the examples which, though not specifically about the element, make use of the element.

The following steps show how to find information about the **Open** method. You can use the same steps for other properties, methods, and events.

### To find information about the Open method:

1. Click the  button in help. The search dialog box opens.
2. Type **Open**. The upper list box scrolls to the Open method entry. You see two entries: "Open method" and "Open method in other examples."
3. Click the entry you want to select it and click the Show Topics button (or double click the one you want). Topics appear in the bottom list box.
4. Click the topic you want to select it and click the Go To button (or double click the topic). The topic appears.

## Related Topics

### Description

[Introducing ABC OLE Automation](#)

[Jumping to Visual Basic Help](#)

[Using the Help System](#)

## Running and Viewing the Sample Files

Related

ABC provides automation samples for both Visual Basic and C++. You can use these samples to get creative ideas for uses of automation, and then examine the code to see how it was done. Each program contains explanatory comments and can be copied or edited to suit your own needs.

To use ABC OLE Automation and to view the code in the samples, you must install Visual Basic 3.0 or later or Visual C++ 1.5 or later. To view the code for the Visual Basic examples, you must install the ABC VB event handler in Visual Basic.

### To install the ABC VB event handler:

1. If necessary, install Visual Basic® 3.0 regular or Professional Edition.
2. Run Visual Basic 3.0.
3. From the Visual Basic menu, open the File menu and choose Add File. The Add File dialog box opens.
4. Switch to the Windows System Directory in the dialog box and select the file ABCAUTO.VBX to install the ABC VB event handler.

### To run an automation sample from the ABC OLE Automation Demos group window in Program Manager:

- Open the ABC OLE Automation Demos group and double click the icon in the window.

### To run an automation sample from the Program Manager:

1. Open the File menu in the Program Manager.
2. Choose Run.
3. Type ABC\SAMPLES\ and the name of one of the ABC Automation samples: DEPLOY.MAK, EVENTS.MAK, EXCEL.MAK, FIELD.MAK, LINEDRAW.MAK, MENU.MAK, MOVE.MAK, ORGCHART.MAK, NETWORK.EXE, or T\_ONLINE.EXE. **Note:** OLE\_VBX.MAK, the C++ sample is in ABC\SAMPLES\OLE\_VBX.

### To run an automation sample from Visual Basic:

1. Run Visual Basic 3.0.
2. From the File menu, choose Open Project.
3. In the Open Project dialog box, switch to the ABC\SAMPLES directory and choose one of the ABC OLE Automation samples: DEPLOY.MAK, EVENTS.MAK, EXCEL.MAK, FIELD.MAK, LINEDRAW.MAK, MENU.MAK, MOVE.MAK, ORGCHART.MAK, NETWORK.MAK, or T\_ONLINE.MAK. The project window will open.
4. From the Visual Basic Run menu, choose Start.

### To view the code in one of the Visual Basic sample files:

1. Run Visual Basic 3.0.
2. From the File menu, choose Open Project.
3. In the Open Project dialog box, switch to the ABC\SAMPLES directory and choose one of the ABC OLE Automation samples: DEPLOY.MAK, EVENTS.MAK, EXCEL.MAK, FIELD.MAK, LINEDRAW.MAK, MENU.MAK, MOVE.MAK, ORGCHART.MAK, NETWORK.MAK, or T\_ONLINE.MAK. The project window will open.
4. From the Visual Basic View menu, choose Code.  
*or*  
Click the View Code button in the project window.  
*or*  
Double click on the .FRM name of project in the project window, and then double click on the button in the form that opens.

**Note:** If ABC is not running when you run one of the samples from Program Manager or the ABC FlowCharter Window, ABC will be started automatically. When you exit ABC, any samples that are not

running will be closed automatically.

ABC OLE Automation includes the following samples.

[Deployment Wizard](#)

[Double Click and Delete Events Demo](#)

[Double Click Line Draw](#)

[Excel 5.0 Data Sample](#)

[Field Change Notify](#)

[Menus Sampler](#)

[Move Event Demo](#)

[Organizational Chart Generator](#)

[VC++/MFC Events Sample](#)

[Network Database](#)

[Text on Lines](#)

## Related Topics

### Description

[Introducing ABC OLE Automation](#)

[Jumping to Visual Basic Help](#)

## Deployment Wizard

Related



The Deployment Wizard is a Visual Basic program that helps make deployment charts. Use the mouse to select the departments and phases you want on the chart. You can add and delete items from the Departments and Phases lists. Any settings you make are saved in DEPLOY.INI and restored when the program is run again.

After the chart is generated, try moving and resizing the boxes that list the Departments and Phases. They have special snapping behavior that is driven by event handling in the Deployment Wizard. (The ABCAUTO.VBX events [ObjectSizedNOTIFY](#), [ObjectSizeSUBCLASS](#), and so forth, trigger the Visual Basic snapping code.)

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Double Click and Delete Events Demo

Related



The Double Click and Delete Events demo alters the double click and **Del** key behavior. When you double click a shape, it turns red and the text "You double-clicked on me!" appears in the shape. When you select a shape and press the **Del** key, it remains on the page and its fill color changes to gray, instead of being deleted.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Double Click Line Draw

Related



The Double Click Line Draw sample draws a line between two shapes after you double click each shape.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Excel 5.0 Data Sample

Related



The Excel 5.0 Data sample reads an Excel 5.0 data file and uses the data to generate field values in a flowchart.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Field Change Notify

Related



The Field Change Notify sample displays a message box when a field is changed in the field viewer. The message box displays the name of the changed field and its contents.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Menu Sampler

Related



The Menu sample adds a menu to ABC called "Stats." This menu has two items that count the objects in the chart. When MENU.EXE shuts down, ABC automatically removes the "Stats" menu.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Move Event Demo

Related

Related

The Move Event demo causes a single shape to turn yellow if it is moved. If you move more than one object, the moved objects turn green. Additionally, fields for the X and Y positions are maintained below each moved shape.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Organizational Chart Generator

Related



The Organizational Chart Generator is a Visual Basic program that makes an ORG chart from a text file. The text file uses tabs to indicate the levels in the organization. Two TXT files (ORGCHRT1.TXT and ORGCHRT2.TXT) are installed in the ABC\SAMPLES directory that you can edit and use to generate organizational charts.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## VC++/MFC Events Sample

Related



The VC++/MFC Events sample is written in C++. (See [Using C++ with ABC OLE Automation](#) for more information.) The sample alters the double click and **Del** key behavior. It turns an object green when it is double clicked. If the object is a shape, the shape's text changes to "C++ is easy!" When you select objects and press the **Del** key, they remain on the page and turn gray, instead of being deleted.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Network Database

Related

Related

Network Database is a Visual Basic program that creates a visual image of the connections of a computer network. After the chart is created, you can double click on a shape for information on that node.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## Text on Lines

Related

Related

Text on Lines is a Visual Basic program that demonstrates how to work with text on lines. The program opens a chart that has three text objects on a line. The user can specify which of the text objects he or she wants to turn blue.

## Related Topics

### Description

[Running and Viewing the Sample Files](#)

## How to Use this Help

Related

This Help gives you information about how to write ABC OLE Automation programs for ABC. Use this Help to learn the highlights of Visual Basic, as well as how to automate the features of the ABC product.

We recommend you take a few minutes to become familiar with this Help and its contents before using ABC OLE Automation. You will find it provides information to help you understand the basic concepts of Visual Basic, detailed information for automating each feature of ABC, and a multitude of examples for each programming property and method.

Refer to [Visual Basic Concepts](#) and [Writing a Program](#) for information to familiarize yourself with Visual Basic. They describe the basic concepts that are used throughout the remainder of the Help and provide useful information on using Visual Basic to write an automation program. Detailed information about Visual Basic is provided in the Visual Basic manuals.

Refer to [Running and Setting Up ABC](#), [Handling ABC Events](#), [Working with Chart Files](#), [Working with Objects](#), [Working with Shapes](#), [Working with Lines](#), [Working with Text](#), [Working with Data Fields](#), and [Using Color](#) to see how to automate the features of ABC. Each topic provides detailed information and examples on how to perform each task using automation commands. Use these topics to learn the details associated with automation. These topics assume you are familiar with Visual Basic and ABC. To learn about ABC, see the *ABC FlowCharter User's Guide*, one of the manuals that came with ABC FlowCharter or the Help for ABC Viewer.

[Language Reference](#) is a complete reference for every property, method, and event used with ABC OLE Automation. After you know the basic concepts behind a property or method, use this as a quick reference for information on syntax and parameters, as well as a description of possible values. Topics below this one include ways to access the properties, methods, and events. [Objects, alphabetical](#) lists all the objects alphabetically. [Objects, graphical](#) provides a visual reference to the ABC objects and their relationships. [Properties, alphabetical](#) lists the properties in alphabetical order. [Methods, alphabetical](#) lists the methods in alphabetical order. [Events, alphabetical](#) lists the events in alphabetical order. [All Properties, Methods, Objects, and Events, alphabetical](#) lists the properties, methods, and events in alphabetical order.

[ABC Menu Command equivalents](#) provides a listing of the ABC OLE Automation command that is equivalent to each ABC menu command.

[ABC Features Not Automated](#) describes the ABC features that cannot be automated.

[Using C++ with ABC OLE Automation](#) contains information on accessing and using ABC OLE Automation with C++.

## Related Topics

### Description

[Controlling ABC FlowCharter and ABC Viewer](#)  
[Introducing ABC OLE Automation](#)

## Conventions

Related

This Help provides visual keys to special information with Notes and Tips.

**Note:** Notes inform you of exceptions or special cases.



Tips offer ways to help you work more efficiently, and suggest shortcuts.

This manual also uses the conventions listed in the following table.

### Example

**Height**, DrawLine

*ShapeObject*

[*FieldType*]

{*Index* | *Filename*}

' This is a comment

ABCObject.Height = 2.5

CONSTANT.AF3

**Enter**, **Del**

### Description

Words in bold or green with an underline indicate properties, methods, and events. You can click on any item that is green with an underline to go to a related topic.

Words in italic indicate placeholders for information you supply.

Elements within square brackets are optional.

Elements within braces separated by a vertical bar represent a mandatory choice between the two elements. You may choose one element or the other.

An apostrophe ( ' ) in code introduces a comment. Comments are ignored by the system when the program runs, but provide helpful information to a person reading the code.

Text in this font represents actual Visual Basic or C++ code.

Words in all capital letters represent file names or constants.

Words in bold with an initial capital letter represent keys you can press on your keyboard.

## Related Topics

### Description

[Introducing ABC OLE Automation](#)

## Accessing ABC OLE Automation from Visual Basic

Related

There are several ways you can access this help from Visual Basic. Each way brings up an appropriate topic.

- Select the ABC Events VBX and press **F1**.
- Highlight a procedure name in the **Procedure box** ("Proc:" combo box) and press **F1**.
- Highlight a property from the **Properties Window** and press **F1**.

## Related Topics

### Description

[Accessing ABC OLE Automation Help from ABC FlowCharter Help](#)

[Accessing ABC OLE Automation Help using Excel 5.0](#)

[Accessing ABC OLE Automation Help using the File Manager](#)

[Jumping to Visual Basic Help](#)

[Searching in Help](#)

[Using the Help System](#)

## Controlling ABC FlowCharter and ABC Viewer

Related

You can use ABC OLE Automation to control both ABC FlowCharter and ABC Viewer. This topic describes how to use ABC OLE Automation, ABC FlowCharter, and ABC Viewer.

Generally, you want to develop ABC OLE Automation programs using ABC FlowCharter. After the program and the ABC FlowCharter files work to your satisfaction, you can apply the program to ABC Viewer, a simpler program that allows entering values in data fields but does not allow creating or moving objects.

For example, suppose you want to provide customers with ABC Viewer and a chart that lists your products and their retail prices. When a customer clicks on an object, you want to provide a dialog box containing information about that product and why they should buy it. You can display the dialog box using ABC OLE Automation.

### Which Program Runs?

When you run an ABC OLE Automation program, it runs either ABC FlowCharter or ABC Viewer. The following rules determine which program runs.

- If either ABC FlowCharter or ABC Viewer is running, then the ABC OLE Automation program applies to it.
- If both ABC FlowCharter or ABC Viewer are running, then the ABC OLE Automation program applies to the program that started second.
- If neither ABC FlowCharter nor ABC Viewer is running, then the ABC OLE Automation program runs the program that ran most recently.

## Related Topics

### Description

[Introducing ABC OLE Automation](#)

[Other Programs in ABC FlowCharter 4.0](#)

Related

## Visual Basic Concepts

Related

[Introduction to Programming](#)

[Objects in Visual Basic](#)

[Properties in Visual Basic](#)

[Methods in Visual Basic](#)

[Events in Visual Basic](#)

## Related Topics

Description

[Contents](#)

## Introduction to Programming

Related

Visual Basic is a variant of the BASIC programming language designed specifically for creating applications for Microsoft Windows. [Related](#). Visual Basic differs from earlier versions of BASIC in two important respects.

- You program Visual Basic in a graphic environment in which many aspects of program development are accomplished by drawing on the screen using a mouse. It is this distinctive characteristic of Visual Basic programming that gives the language the "visual" part of its name and greatly simplifies the process of creating applications.
- Visual Basic is an object-oriented language. Object-oriented programming (OOP) is a relatively new approach to software development.

For more information on the operation and use of Visual Basic's innovative graphic programming environment, see [Writing a Program](#).

This topic and the other topics referred to in the [Visual Basic Concepts](#) topic explain the terms and concepts behind object-oriented programming as implemented in Visual Basic. The terms defined in these topics are used throughout this manual.

## Related Topics

### Description

[Visual Basic Concepts](#)

## Objects in Visual Basic

Related

In an object-oriented language such as Visual Basic, the emphasis of program development is on the definition and use of specialized software units called objects. This is in contrast to other developmental methodologies in which the major emphasis is on the flow of program execution and the actions performed by the parts of a program.

An *object* is a software structure that combines both data and the capability to act upon or process that data.

A Visual Basic object is defined by five characteristics.

- It has a unique name.
- It has data stored within it that defines its state at any given time. The data items of an object are called *properties* in Visual Basic.
- It can perform operations. These operations are called *methods*.
- It can recognize actions such as keystrokes or mouse clicks. The user or system actions that an object recognizes are called *events*. You determine the response of an object to an event through the program instructions that you place in the *event procedure* of the object.
- It has relationships to other objects. The relationships between objects divide the objects into groups called *classes*, which are arranged in a *hierarchy*.

In some object-oriented languages, you can create new types of objects. In Visual Basic, you are limited to the objects provided by Visual Basic plus any custom objects supplied by add-on products such as ABC.

### ABC Objects

ABC exemplifies the power and capabilities of the object-oriented approach to programming. The objects provided by ABC make it easy for a Visual Basic application to create diagram and flowchart shapes, connect the shapes with lines illustrating relationships, and label components.

### Object Hierarchy

For information about the hierarchy of objects in ABC, see [Objects, graphical](#).

## Related Topics

### Description

[Visual Basic Concepts](#)

## Properties in Visual Basic

Related

The data items stored within an object are called *properties*. Each object has specific properties that you can set and control. Some examples of properties are the **BorderColor**, **BorderStyle**, **BorderWidth**, **FillColor**, and **FillPattern** properties of the ABC [Shape](#) object.

You can set some properties to any value, while other properties are restricted to specific values. An example of a property with a restricted value is the **BorderWidth** property of the ABC Shape object. The **BorderWidth** property must be set to a whole number from 1 (thinnest) to 5 (thickest).

## Related Topics

### Description

[Visual Basic Concepts](#)

## Methods in Visual Basic

Related

A *method* is an operation that an object can perform. Some examples of methods are the **Activate**, **CloseChart**, **Copy**, **Cut**, **DrawLine**, **DrawShape**, and **PrintOut** methods of the ABC [Shape](#) object.

The methods available to an object are predefined and depend upon the object.

## Related Topics

### Description

[Visual Basic Concepts](#)

## Events in Visual Basic

Related

An *event* is an action recognized by an object. Events can be triggered by the user (such as clicking an object with the mouse), by the computer system (such as a timer event), or by program code (such as an instruction to move an object).

An event is always specific to a particular object, which means that a given event can be detected by only one object. For example, a mouse click triggers an event only for the object actually clicked. Other objects beside or beneath the clicked object do not recognize that click event.

The events recognized by an object are predefined and depend upon the object. For example, the events that can be recognized by ABC OLE Automation are listed below.

AppMenuHintSUBCLASS	FieldValueChangedNOTIFY
AppMenuPopupSUBCLASS	LinkNOTIFY
AppMenuSUBCLASS	NewLineNOTIFY
AppQuitNOTIFY	NewShapeNOTIFY
AppQuitSUBCLASS	ObjectClickSUBCLASS
ChartActivateNOTIFY	ObjectFontChangeNOTIFY
ChartChangeNOTIFY	ObjectLineAttachNOTIFY
ChartCloseSUBCLASS	ObjectMovedNOTIFY
ChartNewNOTIFY	ObjectMoveSUBCLASS
ChartOpenNOTIFY	ObjectSizedNOTIFY
ChartPasteNOTIFY	ObjectSizeSUBCLASS
DeleteSUBCLASS	ReplaceShapeNOTIFY
DoubleClickSUBCLASS	SpecialKeySUBCLASS
ExclusiveSelectionNOTIFY	

If you want an object to perform a task when an event occurs, you add program instructions to the *event procedure* for that event. When an object detects an event for which it has a defined event procedure, it executes the instructions in the procedure. At the conclusion of the event procedure, the object returns to a state in which it waits for another event.

An example of an ABC event procedure is shown below. This procedure saves the left and top locations of the object before it is moved in the global variables GLeft and GTop.

```
Sub ABC1_ObjectMoveSUBCLASS ( )
    GLeft = ABC1.Object.Left      ' Save left edge
    GTop = ABC1.Object.Top        ' Save top edge
End Sub
```

**Note:** ABC requires that you identify the event procedures you want it to execute by registering them using the [RegisterEvent method](#) of the ABC Application object. Unregistered events are ignored by ABC, even though you may have written procedures for the events. For more information on registering events, see [Registering Event Procedures](#).

### Overriding Standard Behavior

Many events have *standard* actions that they perform when the event is triggered. For example, the standard action of the [ObjectSizeSUBCLASS event](#) is to resize the selected object or objects.

You can cancel the standard action of a SUBCLASS event by setting the **Override** property of ABC1 to True in the object's event procedure.

The following statements provide an example of overriding an event's standard behavior. The standard action of the [DeleteSUBCLASS event](#) is to delete the selected object or objects. If **Override** is set to True

in the [DeleteSUBCLASS event](#) procedure, then the delete action is not performed.

```
Sub ABC1_DeleteSUBCLASS ( )  
    Dim ABCObj As Object  
    Set ABCObj = ABC1.Object  
    ABC1.Override = True           ' Override standard action  
End Sub
```

For more information on overriding standard behavior, see the section [What Are ABC Events?](#)

## Related Topics

### Description

[Visual Basic Concepts](#)

Related

## Writing a Program in Visual Basic

Related

[The Visual Basic Programming Environment](#)

[Creating a Visual Basic Application](#)

[Designing a Form](#)

[Writing Code](#)

[Using Procedures](#)

[Understanding Data Types](#)

[Declaring Object Variables](#)

[Setting and Retrieving Properties with Code](#)

[Executing Methods](#)

## Related Topics

Description

[Contents](#)

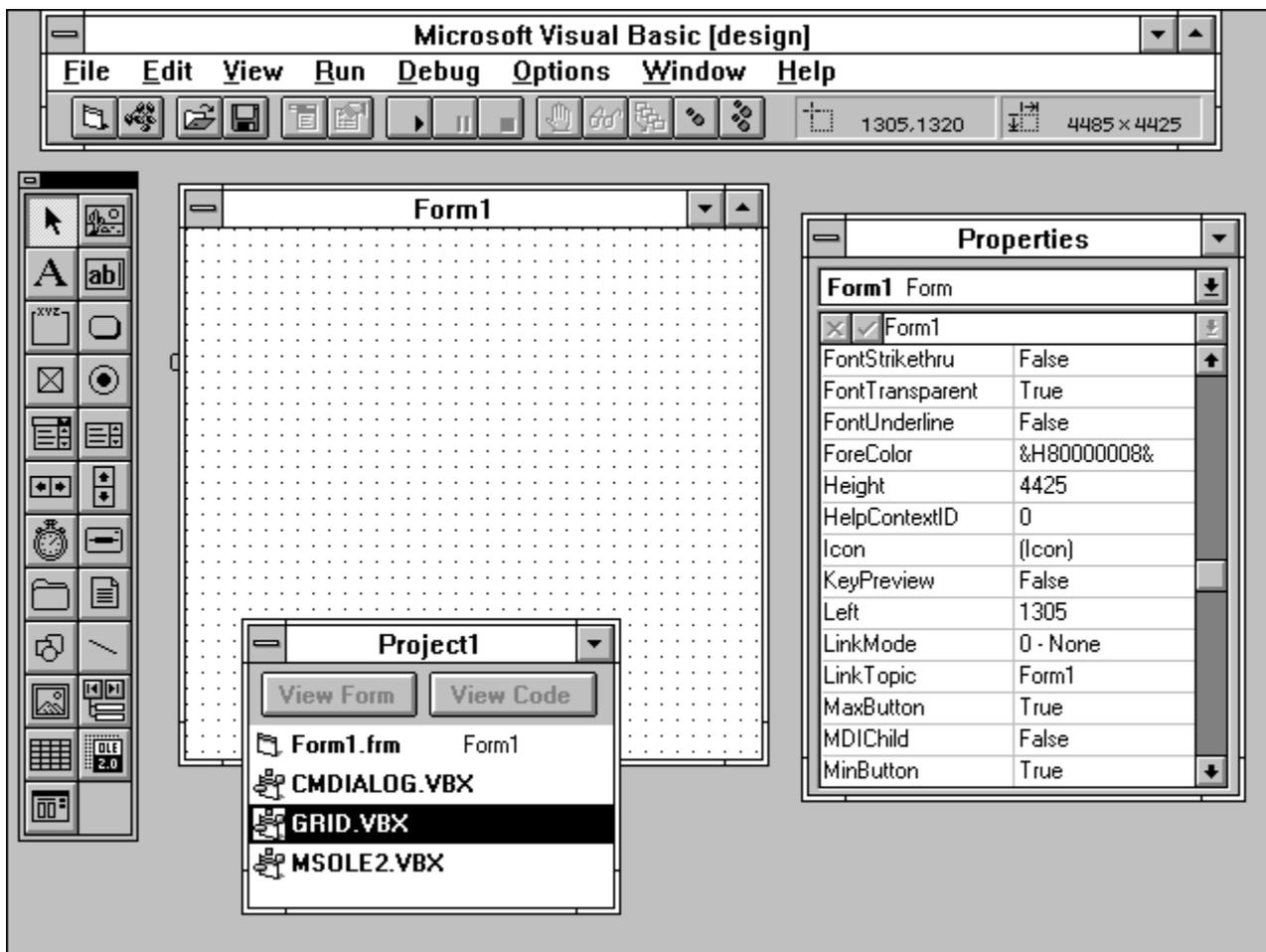
## The Visual Basic Programming Environment

Related

Starting Visual Basic displays the Visual Basic programming environment. As you would expect in a programming language designed to create Windows applications, the Visual Basic programming environment is organized into windows. It consists of five specialized windows. (Not all windows are visible when you start Visual Basic.)

- Main window
- Project window
- Form window
- Toolbox
- Properties window

You can resize and reposition these windows as desired. A typical arrangement in which all of the windows are visible is shown below.



[The Visual Basic Main Window](#)

[The Visual Basic Tool Bar](#)

[The Visual Basic Project Window](#)

[The Visual Basic Form Window](#)

[The Visual Basic Toolbox](#)

[The Visual Basic Properties Window](#)

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

## The Visual Basic Main Window

Related

In Visual Basic, the Main window contains the Visual Basic Control box, title bar, Minimize button, Maximize button, menu bar, and toolbar. The Main window is generally positioned at the top of the screen, but can be moved if desired.



## Related Topics

### Description

[The Visual Basic Programming Environment](#)

## The Visual Basic Tool Bar

Related

The toolbar in the Main window contains buttons providing quick access to common Visual Basic commands. The toolbar also contains two fields giving information on the screen location and size of the selected form or object.



## Related Topics

### Description

[The Visual Basic Programming Environment](#)

## The Visual Basic Project Window

Related

A *project* is the collection of files used to build a Visual Basic application. Visual Basic uses specialized files to store different types of program data. To make it clear that program development in Visual Basic involves a series of files, Visual Basic programs are called projects.

The Project window lists all the files associated with the current project.

The files in a project include the following.

- A file for each form in the application. Form files have the FRM extension.
- A file for each code module in the application. Code files have the BAS extension.
- A file for each custom control used by the application. Custom control files have the VBX extension. For example, the controls provided by ABC are stored in the file ABCAUTO.VBX.

The files and various settings used by a project are saved in the project file. Project files have the MAK extension. The default project file that is loaded automatically when you start Visual Basic is called AUTOLOAD.MAK.

Commands are available in the File menu to add and remove files from the Project window.

### To add the ABCAUTO.VBX custom control file to a project:

1. Open the File menu.
2. Click the Add File command. The Add File dialog box opens.
3. Change to the directory where you installed ABC.
4. Double click the file ABCAUTO.VBX. The dialog box closes and the file is added to the project.



If you installed all options of the Visual Basic 3.0 Professional Edition, there will be many VBX files in your default project. You can shorten the time it takes Visual Basic and your programs to load by removing any VBX files you are not using regularly from the default project file. To remove a VBX file from a project, select the file in the Project window, open the File menu, and choose Remove File. When finished, save the project as AUTOLOAD.MAK using the Save Project As command of the File menu.

Besides the files in the project, the Project window contains the View Form and View Code buttons.

The View Form button lets you display the form saved in a form (FRM) file.

The View Code button lets you display the Visual Basic code saved in a form or code (BAS) file.

To display the form saved in a form file, select the form file and click View Form. To display the code saved in a form or code file, select the form or code file and click View Code.



Double click a form file to display its form, or double click a code file to display its code.

## Related Topics

### Description

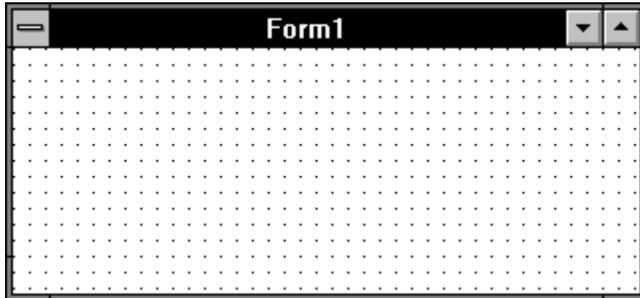
[The Visual Basic Programming Environment](#)

## The Visual Basic Form Window

Related

The first step in creating a Visual Basic application is designing the application's windows. In Visual Basic, during the development stage, windows are referred to as *forms* and are created using the Form window.

When you start a new project, Visual Basic creates an empty form that it titles Form1. Every Visual Basic project must have at least one form, although it is not necessary that the form contain anything or actually be displayed by the completed application.



Notice that even a blank Form window looks like a standard Windows window. It can be resized, and contains a Control box, a title bar, and Minimize and Maximize buttons. The grid within the window is used to align the objects placed on the form. The grid does not appear in the completed application. The procedures for adding objects to the form are discussed in [Drawing a Control](#).

The Form1 title given to the initial form is a default name that Visual Basic supplies automatically for a new project. You can change this name easily to suit the application you are developing.

## Related Topics

### Description

[The Visual Basic Programming Environment](#)

## The Visual Basic Toolbox

The objects placed on a form are called *controls*.

To create a control, you click the appropriate button in the Toolbox and then drag the mouse over the area in the Form window where you want the control. The creation of controls using graphic methods is perhaps the most spectacular aspect of Visual Basic programming.

After you create a control, you define its properties using the control's Properties window. You define the control's operation or action in the completed application by writing Visual Basic code for the control.

Visual Basic comes with a set of standard controls that include labels, picture boxes, text boxes, check boxes, list boxes, command buttons, and timers. The standard Visual Basic and ABC controls are shown below. The Toolbox also has an item for each custom control (VBX) that is added. If you have installed the Professional Edition of Visual Basic, the Toolbox contains many more controls than what is illustrated below. (See the Tip in [The Visual Basic Project Window](#) for instructions on removing these extra controls.) When you add a new control to a project, such as that supplied by ABC, the new control appears in the Toolbox. Therefore, if you are using more than the standard controls, your Toolbox will show the additional controls.

A description of the function of these controls is provided in [Standard Controls](#).

## Related Topics

### Description

[The Visual Basic Programming Environment](#)

## The Visual Basic Properties Window

Related

The Properties window lets you view and set the properties of objects. An object's properties determine its appearance and behavior. The properties of an object include its name, size, screen location, color, and visibility in the completed application.

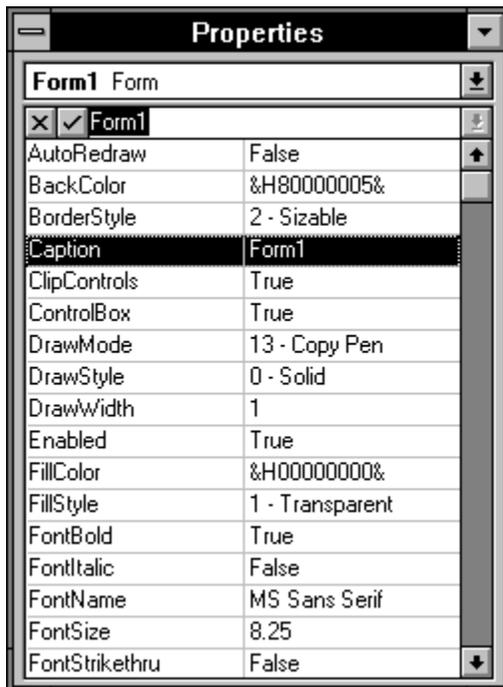
You can activate the Properties window in several ways.

- Click an object such as a form or control.  
or
- Open the Windows menu in the Main window and choose Properties.  
or
- Press **F4**.  
or



- Click the Properties Window button in the Main window toolbar.

The Properties window contains an Object box, a Settings box, and a two-column list of the properties and settings.



The Object box displays the name and type of the object whose properties are listed. The object's name is bold and listed first. In the example above, the object is named Form1 and is a Form object. To the right of the Object box is a down arrow. Click this arrow to display a list of all of the objects associated with the currently selected form, including the Form object itself. Click the appropriate entry in the Object list to display the properties of that object.

You use the Settings box to change the value of a property. The way you change a value depends on whether the property is an enumerated or nonenumerated property.

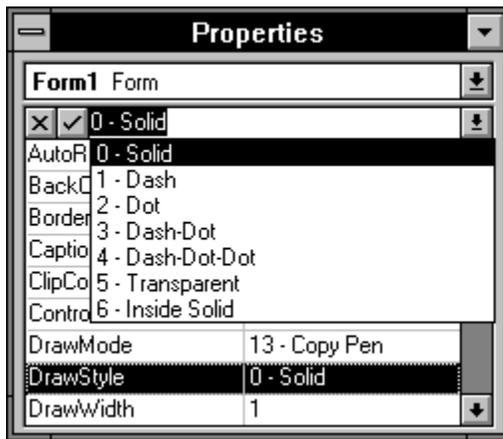
- An *enumerated* property is limited to a set of predefined values. An example of an enumerated property is the **BorderStyle** property of a Form object. **BorderStyle** must be set to None, Fixed Single, Sizable, or Fixed Double. When a selected property is enumerated, you can click the down

arrow (or ellipsis for color properties) to the right of the Settings box to display the list of predefined values.

- A *nonenumerated* property is not limited to a set of predefined values. An example of a nonenumerated property is the **Name** property of an object, which can be set to any text string. When a selected property is nonenumerated, the down arrow to the right of the Settings box is gray.

### To change the value of an enumerated property:

1. Select the property by clicking it. The selected property is highlighted and the current value of the property appears in the Settings box.
2. Click the down arrow to the right of the Settings box and select the desired value from the list that appears. (If the property is a color rather than a setting, click the ellipsis to the right of the Settings box and select the desired color from the color palette that appears.)



### Related

You can cycle through the predefined values of an enumerated property by double clicking the property in the Properties list. For example, to change the **Visible** property of a Form object from True to False, double click Visible in the Properties box.

### To change the value of a nonenumerated property:

1. Double click the property you want to change. The selected property is highlighted and the current value of the property is highlighted in the Settings box.

2. To replace the current property value, just type the new value. The characters that you type appear in the property list and the Settings box.

or

To edit the current property value, click the highlighted value in the Settings box and edit the value as you would any field.



3. To accept the new property value, click the checkmark box in the Settings box or press **Enter**.

or



To reject the new property value, click the X box in the Settings box or press **Esc**.

## Related Topics

### Description

[The Visual Basic Programming Environment](#)

## Creating a Visual Basic Application

Related

The general steps to writing a Visual Basic application are:

1. Create a new project.
2. Design the application's forms (windows).
3. Add code to determine the application's response to desired events.
4. Run and test the application.
5. Create an executable (EXE) file.

These steps are discussed below.

### Creating a New Project

The first step in creating a Visual Basic application is to create a new project. To perform this simple but important step, open the File menu in the Visual Basic Main window and choose New Project. When you create a new project, you ensure that your application begins with a "clean slate."

If you want to use controls in your application that are not part of the standard Visual Basic set, you must add the VBX files that define the controls to your project. For example, to use the control provided by ABC, you must add ABCAUTO.VBX to your project. See [The Visual Basic Project Window](#) for instructions on adding ABCAUTO.VBX to a project.

### Designing the Application's Forms

The actual programming of an application begins with the design of the forms for the application. This is the visual user interface of the application. The process of designing a form is explained in [Designing a Form](#).

### Adding Code

After you create the visual interface, you add code to specify how the program should respond to actions by a user. For example, you add code to a Command button control to determine what the program does when a user clicks the Command button. The process of adding code to an application is discussed in [Writing Code](#).

### Running the Application

Running your application in the Visual Basic programming environment lets you test the application as you develop it.

#### To run an application:



- Click the Start button in the Main window toolbar.  
*or*
- Open the Run menu in the Main window and choose Start.  
*or*
- Press **F5**.

#### To stop a running application:



- Click the End button in the Main window toolbar.  
*or*

- Open the Run menu and choose End.

### **Creating an EXE Program**

The final step in creating a Visual Basic application is to make an executable file. An executable file is a version of the application that does not require the Visual Basic programming environment to execute (it runs as a standard Windows application). To make an executable file for the application, open the File menu in the Visual Basic Main window and choose Make EXE File.

After you make an executable file for your application, you can create a button for it in ABC FlowCharter. For more information, see [Adding Buttons](#).

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

## Designing a Form

Related

Visual Basic programming begins with the creation of forms. The forms you create become windows in the completed application.

The process of designing a form involves setting the properties of the form, drawing the controls that appear on the form, and setting the properties of the controls.

If your application is a simple one, then you may not have more than one form. If you need additional forms, add them with the New Form button in the toolbar or with the New Form command in the File menu.

[Setting a Form's Properties](#)

[Standard Controls](#)

[Drawing a Control](#)

[Resizing a Control](#)

[Moving a Control](#)

[Deleting a Control](#)

[Copying or Cutting a Control](#)

[Aligning Controls](#)

[Setting Control Properties](#)

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

## Setting a Form's Properties

Related

When you start a new project, you are presented with a blank form titled Form1. The properties of this form (or any new form) are set to default values.

To change the form's appearance and behavior, you must customize its properties. You can customize the properties of a form by manually setting the properties using the Properties window, by setting the properties using code, or by using a combination of manual settings and code (the usual approach).

Generally, you set the properties controlling a form's overall appearance manually. These properties include the form's **Caption**, **BackColor**, **ForeColor**, and **BorderStyle**. If you intend to change the **Name** of a form, you should also set this property manually. (The **Name** of a form is used to reference the form in code, so changing a form's **Name** after you have already written code requires replacing references to the old form **Name** in the code with the new form **Name**.)

The process of setting a form's properties manually is discussed in [The Visual Basic Properties Window](#).

The process of setting a form's properties with code is discussed in [Setting and Retrieving Properties with Code](#).

## Related Topics

### Description

[Designing a Form](#)

## Standard Controls

Related

The Visual Basic Toolbox provides standard controls that can be placed on a form. This section gives a brief description of the functions of these controls.



The picture box control displays graphic images from bitmaps, icons, or Windows metafiles. The size of the picture box determines how much of the graphic image is displayed. To make a picture box automatically resize itself so that it can display all of a graphic image, set the control's **AutoSize** property to True.



The label control displays text on a form that the user cannot change. To set a label control to display multiple lines, set the control's **AutoSize** and **WordWrap** properties to True.



The text box control provides an area on a form that can display text and accept user input. To set a text box so it can display and accept multiple lines, set the text box's **MultiLine** property to True.



The frame control provides a way to group controls. The grouping capability provided by a frame is visual and logical (functional).

- A frame groups controls *visually* by surrounding the controls with a box and providing a title.
- A frame groups controls *logically* by permitting an application to treat the controls in the Frame as a unit. For example, setting a frame's **Visible** property to False makes all of the controls in the frame invisible.

To place a control in a frame, you must either draw the control in the frame or Paste it into the frame. You cannot move a control into a frame.



The command button control displays a button that can begin, interrupt, or end a process. The user executes a command button by clicking it. To enable a command button to be executed by pressing enter, set the control's **Default** property to True. Examples of command buttons are the OK and Cancel buttons you see in Windows applications.



The check box control displays a box that can be selected or deselected, like an on/off switch. A check box control shows an X when selected. How the states of a check box are interpreted by an application depend on the code. Check boxes are commonly used for Yes/No and True/False options.



The option button control displays a button that can be selected or deselected. Option buttons, like check boxes, are on/off switches. Unlike check boxes, however, only one option button on a form can be selected at a time. Clicking one option button automatically deselects all other option buttons on the form.

If you want more than one set of option buttons on a form, then you must group them with a frame control.



The combo box control displays a box that combines the characteristics of a text box and a list box. A combo box lets a user select an item from a list by typing the desired item or by choosing the item from a drop-down list.



The list box control displays items that can be selected by clicking the item or by moving the cursor to the item and pressing enter. Scroll bars are automatically added to a list box if it contains more items than can be listed at once. The control's **Additem** and **Removeitem** methods are used to add and delete list items.



The horizontal scroll bar control provides a method of scrolling horizontally through a list or graphically indicating settings such as quantity or volume.



The vertical scroll bar control provides a method of scrolling vertically through a list or graphically indicating settings.



The timer control executes events at specified time intervals. A timer control can be seen when you are designing a form, but is invisible when the application executes.



The drive list box control provides a method of selecting a disk drive when an application executes. This control automatically lists all the valid drives in the user's system.



The directory list box control displays directories and paths when an application executes and provides a method of selecting a directory.



The file list box control displays files in a directory when an application executes, and provides a method of selecting a file. The control's **Path** property determines the directory displayed. The control's **Pattern** property determines the files displayed.



The shape control displays a rectangle, square, oval, circle, rounded rectangle, or rounded square. The control's **Shape** property determines its shape.



The line control displays a horizontal, vertical, or diagonal line. The line control's **BorderStyle** property determines the type of line (solid, dotted, dashed).



The image control displays bitmaps, icons, or Windows metafiles. An image control uses fewer system resources and displays faster than a picture control. It also offers fewer properties and methods than a picture control. Set the control's **Stretch** property to True if you want the image resized to fit the control's size and shape.



The data control provides access to data stored in databases. The data control's **DatabaseName** property determines the database source file.



The grid control displays a grid of cells that can contain text or pictures. The control's **Cols** and **Rows** properties determine the number of rows and columns. **Note:** The grid control is available only if you add the GRID.VBX control file to a project.



The OLE control provides a method of linking and embedding an OLE object into an application. **Note:** The OLE control is available only if you add the MSOLE2.VBX control file to a project.



The common dialog control provides standard dialog boxes for operations such as opening, saving, and printing a file. A common dialog control is shown only as an icon when you are designing a form. It appears as a full-size dialog box when the application executes. **Note:** The common dialog control is available only if you add the CMDIALOG.VBX control file to a project.



The ABC control lets you add ABC OLE Automation capabilities to your Visual Basic applications. **Note:** The ABC control is available only if you add the ABCAUTO.VBX control file to a project.

## Related Topics

### Description

[Designing a Form](#)

## Drawing a Control

Related

There are two ways to draw a control on a form: the normal drawing method and a shortcut method.

### To draw a control using the normal method:

1. Click the appropriate control tool in the Toolbox. For example, to draw a command button, click the Command Button tool.
2. Position the mouse pointer on the form where you want the control to begin. The mouse pointer shows as a cross hair.
3. Press and hold the mouse button, and drag the pointer until the control's outline is the size you want. Release the mouse button. The control appears on the form in the size and shape you drew.

### To draw a control using the shortcut method:

1. Click the form to which you want to add the control. (If there is only one form in your project, you can skip this step.)
2. Double click the appropriate control tool in the Toolbox. The control appears in the center of the selected form.
3. [Resize](#) (and [move](#)) the control as desired.

## Related Topics

### Description

[Designing a Form](#)

## Resizing a Control

Related

Resizing a control is easy and can be performed at any time during the design process.

### To resize a control:

1. Select the control you want to resize by clicking it. Small, solid *handles* appear on the control.
2. Position the pointer over a handle. The pointer changes to a two-headed arrow.
3. Press and hold the mouse button, and drag the pointer in the appropriate direction to change the control's size. Release the mouse button.

## Related Topics

### Description

[Designing a Form](#)

## Moving a Control

Related

You can move controls one at a time or in groups.

### To move a single control:

1. Select the control you want to move by clicking it. The control's handles appear.
2. Position the mouse pointer anywhere on the control other than on a handle.
3. Press and hold the mouse button, and drag the control to its new position. Release the mouse button.

By moving controls in a group, you can keep the relative position of the controls constant.

### To move multiple controls:

1. Select the first control by clicking it. The control's handles appear.
2. Select the additional controls by pressing **Ctrl** while you click the control.
3. Position the mouse pointer on one of the selected controls.
4. Press and hold the mouse button, and drag the controls to their new positions. Release the mouse button.

## Related Topics

### Description

[Designing a Form](#)

## Deleting a Control

Related

To delete a control, select the control and press **Del**. You also can delete a control by selecting the control and choosing the Delete command in the Edit menu.

## Related Topics

### Description

[Designing a Form](#)

## Copying or Cutting a Control

Related

You can copy or cut a control, and then paste it back into your form or into other forms. Select the control and choose the Copy or Cut command in the Edit menu, or select the control and press the shortcut keys **Ctrl+C** (Copy) or **Ctrl+X** (Cut).

## Related Topics

### Description

[Designing a Form](#)

## Aligning Controls

Related

The grid that displays on forms during the development process makes it easy to align controls. As you draw, position, and resize a control, the edges of the control snap to the nearest grid position.

The precision of the grid depends on Visual Basic's Grid Width and Grid Height settings. The default settings for Grid Width and Grid Height is 120 twips, which translates into a distance between grid positions of 1/12 inch. A twip is equivalent to 1/1440 inch (1 inch = 1440 twips).

To increase the alignment precision of the grid, use a larger Grid Width or Grid Height setting. To decrease the alignment precision, use a smaller Grid Width or Grid Height setting.

### To change Visual Basic's grid settings:

1. Open the Options menu in the Main window and choose Environment.
2. Click the Grid Width option (scroll through the list if necessary) and type a value in the Setting box.
3. Click the Grid Height option (scroll through the list if necessary) and type a value in the Setting box.
4. Click OK to accept the new settings and close the dialog box.

If you don't want to use the grid alignment feature, you can turn it off by opening the Environment Options dialog box and setting Align To Grid to No. If you want to hide the grid, open the Environment Options dialog box and set Show Grid to No.

## Related Topics

### Description

[Designing a Form](#)

## Setting Control Properties

Related

After you draw a control, you can customize its appearance and behavior by setting its properties.

As with forms, you probably want to set some control properties manually and some control properties with code. Generally, you set the appearance properties manually. For details on setting control properties with code, see [Setting and Retrieving Properties with Code](#).

### To change a control's properties manually:

1. Select the control as the current object by clicking it. The control's handles appear.
2. The Object box of the Properties window should show the name of the control. If the Properties window is not visible, press **F4** to display it.
3. Set the properties you want to change.

## Related Topics

### Description

[Designing a Form](#)

## Writing Code

Related

[Adding Code to Controls](#)

[Adding Code to Forms](#)

[Writing Code in Event Procedures](#)

[Writing Code in General Procedures](#)

[Writing Startup Code](#)

## Related Topics

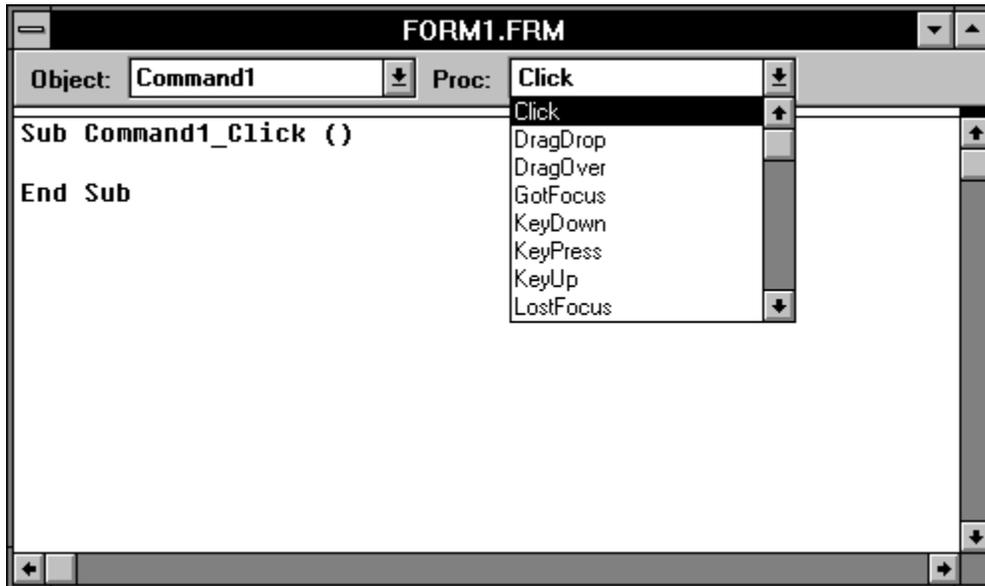
### Description

[Writing a Program in Visual Basic](#)

## Adding Code to Controls Related

Drawing a control and setting its properties let you determine how the control appears in an application, but not what it does. To prescribe what the control does, you must add Visual Basic instructions to the control. These instructions are referred to as *code*.

You add code to a control using the Code window. To display the Code window for a control, double click the control. An example of the Code window for a Command Button appears below.



The title bar at the top of the Code window identifies the control's form. Below the title bar are the Object and Procedure boxes, and below these boxes is the code area.

The Object box gives the name of the object to which the code applies; such as Command1. To select other objects on the form, including the Form object, click the down arrow to the right of the box.

**Note:** Besides listing the objects associated with the form, the Object box list also shows a (general) entry. This section of the form is used to write general procedures for the form, as explained in [Writing Code in General Procedures](#).

The Procedure box gives the name of the event to which the code applies, such as the **Click** event. To select other events defined for this object, click the down arrow to the right of the box. The two statements shown in the code area are required at the beginning and end of the event code, and are supplied automatically by Visual Basic. Any code that you want to add to an event must be typed between these two statements.

## Related Topics

Description

[Writing Code](#)

## Adding Code to Forms Related

Forms, like controls, require code to determine how they respond to events. To add code to a form, double click the form to display the form's Code window (or select the form's name from the Object box list if the Code window is already open). The events recognized by the form are displayed in the Procedure box list.

## Related Topics

### Description

[Writing Code](#)

## Writing Code in Event Procedures

Related

Visual Basic forms and controls are event-driven objects. This means that forms and controls respond to specific user actions such as being clicked, double clicked, dragged, or scrolled. The events to which a form or control responds are predefined and depend on the type of object. For example, a Command button can be clicked, but not scrolled, while a Scroll bar can be scrolled, but not clicked.

Because controls and forms are event-driven, they can execute code only when an event recognized by the object occurs. Until such an event occurs, an object is inactive.

The code that is executed when an event occurs is called an *event procedure*. For details on the syntax required for procedures, see [Using Procedures](#).

Events can occur only when an application is running, not during program development. Thus, clicking a command button that you have just drawn on a form does not cause a Click event for the control. You must click the command button when the program is executing to trigger the Click event.

Related

You can quickly determine which event procedures have code added to them by checking the drop down list in the Procedure box. Event procedures with added code appear in bold in the list.

## Related Topics

### Description

[Writing Code](#)

## Writing Code in General Procedures

Related

In addition to the event procedures that are triggered by a form or control event, a Visual Basic application can have *general procedures*. A general procedure is any procedure that is not an event procedure.

General procedures let you write Visual Basic code that is not attached to specific events. Because general procedures are not attached to specific events, they can contain common code that is needed by several event procedures. Without a general procedure capability, you would have to write duplicate code for each event procedure that needed to perform a common action.

As an example of the benefits of general procedures, consider an application that has command buttons labeled First Record, Next Record, Previous Record, and Last Record. Rather than duplicating the code required to read a record in each command button's Click event procedure, you could write a general procedure that gets a record based on a parameter passed to the procedure. Then, the Click event procedure for each command button calls the general procedure with the appropriate parameter to read a record.

To ensure that a general procedure is not attached to a specific event, you must write the procedure either in the (general) section of a form or in a code module.

- If you want the general procedure to be available only to controls on a particular form, write the procedure in the (general) section of that form.
- If you want the general procedure to be available to any control on any form in the application, write the procedure in a code module.

### To access the (general) section of a form:

1. Double click the form to display the Code window for the form. The Object box shows the name of the form.
2. Click the down arrow to the right of the Object box. The object list opens.
3. Click the (general) entry in the object list.

### To access a code module:

- To create a code module use the New Module command in the File menu.
- To add code to a code module or to view code already saved in a code module, double click the module's name in the Project window.

## Related Topics

Description

[Writing Code](#)

## Writing Startup Code

Related

A Visual Basic application must begin with some initial event or general procedure. The default setting for an application is to begin with the Load event of the first form of the application. The Load event of a form displays the form. The first form of an application is the form named Form1. Changing the name of this form does not change its status as the first form of the application.

If you want an application to begin with the Load event of another form or with a general procedure, you can change the startup setting. If it is necessary for an application to perform initialization code when it begins, the initialization code must be executed by the specified startup event or procedure.

### To change the startup setting:

1. Open the Options menu and choose Project. The Project Options dialog box displays.
2. Double click the Start Up Form option to open the list of startup settings.
3. Select the form that you want as the startup form.  
*or*  
Select Sub Main if you want the application to begin with a general procedure.
4. Click OK to accept your changes and close the dialog box.

Selecting Sub Main tells Visual Basic to start execution with a general procedure named Sub Main. This procedure must be located in one of the application's code modules.

For information on the startup code required for ABC, see [Starting ABC](#).

## Related Topics

Description

[Writing Code](#)

## Using Procedures

Related

All Visual Basic programming instructions must be written as procedures. *Procedures* are organizational structures designed to promote clear, orderly, and logical programs.

Visual Basic recognizes two types of procedures.

- Sub procedures
- Function procedures

Sub procedures are further divided into two categories, event procedures and general procedures.

[Sub Procedures](#)

[Function Procedures](#)

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

## Sub Procedures

Related

Sub procedures begin with a Sub statement and end with an End Sub statement. These statements are located on separate lines. The instructions to be executed by the procedure are located between the Sub and End Sub statements.

```
Sub ProcedureName (ArgumentList)
    Statements
End Sub
```

The name of the procedure follows the Sub statement. The argument list for the procedure follows the procedure name. If the procedure has no argument list, it still includes an empty set of parentheses.

When a procedure is executed, or *called*, the statements in the procedure are executed.

The rules for naming a Sub procedure depend upon whether the procedure is an event procedure or a general procedure. The name of an event procedure follows the format *object\_event*, where *object* is the name of a form or control and *event* is the name of the event that triggers the procedure. For example, the name of the procedure for the Click event of a command button control named Command1 is Command1\_Click. A Sub statement with the appropriate name is automatically provided by Visual Basic when you open the Code window for an event.

The names of general procedures begin with a letter; can contain only letters, numbers, and underscore ( \_ ) characters; and can be no longer than 40 characters. General procedure names have one other restriction: they cannot be words already used by Visual Basic, such as Beep, Loop, For, If, and Line.

The argument list of a procedure passes values to the procedure that can be used by the statements in the procedure. The syntax for each argument in the argument list is shown below.

```
[By Val] ArgumentName [( )] [As Type]
```

The By Val option determines whether the argument is passed by value or reference. If By Val is used, then the argument is passed *by value*, which means that a copy of the argument is passed to the procedure. Consequently, any change to the argument by the procedure does not change the original. If By Val is omitted, then the argument is passed *by reference*. When an argument is passed by reference, any change to the argument by the procedure changes the original.

The ( ) option specifies that the argument is an array.

The As Type option specifies the data type of the argument. If no data type is specified, the argument defaults to the Variant type. See [Understanding Data Types](#) for details on data types.

An example of a general Sub procedure is shown below. This example reads a record from a file. The file is identified by the value passed to the procedure as FileNum. The record is identified by the value passed as Rec, and the data obtained from the record is stored in Rdata.

```
Sub GetRec (FileNum as Integer, Rec as Long, Rdata as String)
    Get FileNum, Rec, Rdata
Sub End
```

An example of how this procedure can be called is shown below. It calls the GetRec procedure to read record 123 of file 2. The contents of the record is returned in TestResult.

```
GetRec 2,123,TestResult
```

## Related Topics

### Description

[Using Procedures](#)

## Function Procedures Related

Function procedures begin with a Function statement and end with an End Function statement. The instructions executed by the function are located between these two statements.

```
Function ProcedureName (ArgumentList) [As Type]  
    Statements  
    ProcedureName = FunctionResult  
End Function
```

The rules for naming a Function procedure and defining its argument list are the same as for Sub procedures.

The essential difference between a Function and a Sub procedure is that a Function procedure assigns a value to its name. This action must be performed somewhere in the function, as indicated by the *ProcedureName = FunctionResult* line in the syntax format.

Look at the following example of a Function procedure, which converts a Fahrenheit temperature to Celsius.

```
Function Celsius (Fahrenheit as Double)  
    Celsius = (Fahrenheit - 32) * 5/9  
End Function
```

```
T = Celsius (22) * 9 / 5 + 32
```

## Related Topics

### Description

[Using Procedures](#)

## Understanding Data Types

Related

Visual Basic supports seven data types. A *data type* is a standardized method of representing data. The following table provides the names, a brief description, and the range of the Visual Basic data types.

Type	Description	Range
<b>Integer</b>	2-byte integer	-32,768 to 32,767
<b>Long</b>	4-byte integer	-2,147,483,648 to 2,147,483,647
<b>Single</b>	4-byte floating-point number	-3.402823E38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E38 for positive values
<b>Double</b>	8-byte floating-point number	-1.79769313486232E308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232E308 for positive values
<b>Currency</b>	8-byte fixed-decimal number	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
<b>String</b>	String of characters	0 to approximately 65,500 characters
<b>Variant</b>	Variable	Depends upon value stored

The default data type is Variant, so if no data type is declared, then Variant is assumed. If you want a variable or data value to be a different data type, then you must explicitly declare it as that data type.

You can declare a variable's data type in the argument list of a procedure or by using the *As Type* option with variable definition statements such Dim, Static, or Global.

Some examples of these data type declaration methods are shown below.

```
Dim Count As Integer
Dim Employee As String, State As String, JobNumber As Long
Global Total As Currency
```

[Numeric Data Types](#)

[String Data Type](#)

[Variant Data Type](#)

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

## Numeric Data Types Related

Although Integer, Long, Single, Double, and Currency are all numeric data types, they differ in important characteristics such as storage requirements, range, and accuracy.

If you know that a variable will always be a whole number, then consider defining the variable as an Integer or Long data type. Using these data types when possible reduces a program's memory requirements and speeds up its execution.

If a variable can have a decimal value, then define the variable as a Single, Double, or Currency data type. The Single and Double data types have larger ranges than the Currency type, but may have small rounding errors.

## Related Topics

### Description

[Understanding Data Types](#)

## String Data Type

**Related**

Nonnumeric data, such as text, is referred to as *string* data. To enable a variable to store string data, declare it a string variable using the String data type.

Unless you specify otherwise, a string variable can store string data of any size up to approximately 65,500 characters. If you want to limit a string variable to a predefined, fixed size, then use the following syntax to declare the string variable.

As String *Size*

The example below uses this format to declare a string that is fixed at 35 characters in size. If the data stored in `Company` is less than 35 characters, it is padded with trailing spaces up to the 35-character length. If the data stored in `Company` is more than 35 characters, it is truncated from the end to the defined length.

```
Dim Company As String 35
```

## Related Topics

### Description

[Understanding Data Types](#)

## Variant Data Type

**Related**

The default data type is Variant. Variables defined as Variant can store data in Integer, Long, Single, Double, Currency, and String formats, plus a Date/Time format.

When a Variant variable is assigned a value, Visual Basic stores the assigned data in the most efficient data format. For example, if the value assigned to a Variant variable is a whole number between -32,768 and 32,767, then the value is stored as a two-byte integer. If a decimal value is added to this variable, then Visual Basic automatically performs the conversion necessary to store the new value as a floating-point number. If the variable is later treated as a string variable, then Visual Basic converts and stores the variable's value as a string.

You can determine the current storage format of a Variant variable with the **VarType** function. VarType returns a value that indicates the current data format of a variable. The values returned by **VarType** are defined below.

Return Value	Meaning
0	Empty
1	Null
2	Integer
3	Long
4	Single
5	Double
6	Currency
7	Date/Time
8	String

The Empty and Null return values indicate special states. The Empty value indicates that a Variant variable has never been assigned a value. The Null value indicates that a Variant variable does not contain a valid value. For a variable to return a Null value, it must have been explicitly assigned a Null value.

## Related Topics

### Description

[Understanding Data Types](#)

## Declaring Object Variables

Related

A powerful feature of Visual Basic is its capability to assign objects to variables. Object variables let a Visual Basic application manipulate objects as easily as it manipulates string or numeric data.

Before you can use an object variable in your code, you must declare it using the Dim, Static, or Global statement. The general syntax for declaring an object variable is shown below.

```
Dim VariableName As [New] ObjectType  
Static VariableName As [New] ObjectType  
Global VariableName As [New] ObjectType
```

*VariableName* gives the name of the object variable. *ObjectType* determines whether the object variable is declared for a specific or generic object. Specific object variables are declared by giving the **Name** property of the object as *ObjectType*. Generic object variables are declared by giving the class (such as Form or Control) of the object as *ObjectType*.

Adding the New parameter to a declaration statement creates a new object that is identical to the object specified as *ObjectType*. When New is omitted, the declaration statement refers to an existing object.

Some examples of object variable declarations are shown below.

```
Dim InputForm As Form3  
Dim NameBox As New Text1  
Global AppForms As Form  
Dim AppControls As Control
```

The first two examples declare specific object variables. The last two examples declare generic object variables.

An example of declaring an object variable as an ABC object is shown below.

```
Dim ObjectIn As Object
```

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

## Setting and Retrieving Properties with Code

Related

The properties of forms and controls can be set and retrieved in code using assignment statements. This enables a program to change properties based upon calculations or data input from users, files, and other sources.

[Setting Form Properties](#)

[Setting Control Properties](#)

[Setting ABC Object Properties](#)

[Retrieving Properties](#)

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

## Setting Form Properties

Related

The general syntax for setting the property of a form with code is

*Form.Property = Value*

where *Form* is the name of the form and *Property* is the name of the property. Some examples of setting a form's properties are shown below.

```
Form1.BorderStyle = 1  
Form3.Width = 5000
```

If the code setting the property of a form is located on the form, then the form name can be omitted from the assignment statement, as illustrated below.

```
BorderStyle = 1  
Width = 5000
```

## Related Topics

### Description

[Setting and Retrieving Properties with Code](#)

## Setting Control Properties

Related

The general syntax for setting the property of a control with code is

*Form!Control.Property = Value*

where *Form* is the name of the form, *Control* is the name of the control, and *Property* is the name of the property. Some examples of setting a control's properties are shown below.

```
Form1!Command.Caption = "OK"  
Form3!Label2.Visible = 1
```

If the code setting the property of a control is located on the same form as the control, then the name of the form can be omitted from the assignment statement, as illustrated below.

```
Command.Caption = "OK"  
Label2.Visible = 1
```

## Related Topics

### Description

[Setting and Retrieving Properties with Code](#)

## Setting ABC Object Properties

Related

The general syntax for setting the property of an ABC object with code is

*Object.Property = Value*

where *Object* is the name of the ABC object and *Property* is the name of the property. Some examples of setting ABC object properties are shown below.

```
ABC.Visible = True  
Object.Shape.FillColor = ABC.BLUE
```

## Related Topics

### Description

[Setting and Retrieving Properties with Code](#)

## Retrieving Properties

Related

You can retrieve a property setting by naming the property on the right side of an assignment statement. You can use this feature to save the property setting in a variable or to assign it to another property.

The rules for naming a property to be retrieved are the same as for setting a property.

The examples below illustrate various methods of retrieving properties. In the first three examples, the property settings are retrieved and saved in variables. In the last two examples, the setting retrieved from one property is used to set another property.

```
Temp = Form2.Width  
Tstate3 = Form3!Timer1.Enabled  
Command2.Visible = Command1.Visible  
Form5.BackColor = Form1.BackColor
```

## Related Topics

### Description

[Setting and Retrieving Properties with Code](#)

## Executing Methods

Related

Visual Basic objects can perform various actions such as moving and printing data. The actions that an object can perform are called *methods*. The methods available to an object are predefined and depend upon the object.

The general syntax for executing a method with code is shown below. *Object* is the name of a form or control and *Method* is the name of the method to perform. If a method requires parameters, they must follow the *Object.Method* name.

*Object.Method* [*Parameters*]

The following example executes a form method. The **Cls** method clears a form of all graphics and text.

```
Form2.Cls
```

The following example executes a control method. The **SetFocus** method selects the specified text box as the current object and places the cursor in the text box.

```
Text1.SetFocus
```

The following example executes an ABC object method. The [PrintOut method](#) prints the Chart object to the current printer.

```
Chart.PrintOut
```

## Related Topics

### Description

[Writing a Program in Visual Basic](#)

Related

## Running and Setting Up ABC

Related

[Starting ABC](#)

[Controlling the ABC Window](#)

[Getting ABC System Information](#)

[Customizing Preferences](#)

[Setting Defaults](#)

[Customizing ABC](#)

[Displaying Help](#)

[Closing ABC](#)

[Window Handles](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Starting ABC

Related

Starting ABC is the first step in writing an automation program. You define ABC as an object using **Dim**. In Visual Basic, first, enter the following statement in the (declarations) Proc of the (general) Object to make ABC a global variable.

```
Dim ABC As Object
```

You run the application using **Set** and **CreateObject**. To make the application visible, you set the Visible property of the Application object to True.

**Note:** You use these statements, along with the Dim, at the beginning of all ABC OLE Automation programs.

Enter the following statements in the Load Proc of the Form Object.

```
Sub Form_Load ()
    Set ABC = CreateObject("ABCFlow.application")
    ABC.Visible = True
End
End Sub
```



**Note:** If you do not want the program to end after ABC runs, omit the End statement.

The statement `Set ABC = CreateObject("ABCFlow.application")` runs ABC. When you run the program, ABC runs invisibly, if it is not yet running. If it is already running, the statement has no effect, other than to establish a valid ABC application object for use.

The statement `ABC.Visible = True` makes ABC visible.

The statement `End` ends the program.

Related

You might want to save this program as RUNABC so you can open it as the beginning of your other programs. Be sure to save it with a different name before you make changes to it.

Run the program you have written so far from within Visual Basic. You see ABC run. Then the Basic program comes back to the front. If you make the program into an EXE file and then run it, the program runs and stays in front.

**Note:** If ABC is already running, CreateObject does not run a new copy of ABC. Instead, it returns a valid ABC application object for the ABC application that is already running.

## Showing and Hiding ABC

To show ABC, you use this statement.

```
ABC.Visible = True
```

To hide ABC, you use this statement.

```
ABC.Visible = False
```

If you set [Visible](#) to False, the application is still running, but it is not visible. You cannot switch to it using **Alt+Tab**, and it is not shown in the Task List dialog box that appears when you press **Ctrl+Esc**. The default for [Visible](#) is False, so you should always set it to True immediately after you run ABC.

You can use [Visible](#) on the right side of a statement and in conditions. The following statement sets the Boolean variable ABCShown to True or False according to the value of ABC.Visible.

```
ABCShown = ABC.Visible
```

The following statements beep your computer's speaker if ABC is running but not visible.

```
If Not ABC.Visible Then
    Beep
End If
```

## Related Topics

### Description

[Running and Setting Up ABC](#)

## Controlling the ABC Window

Related

You can control the appearance of the ABC window using many different properties and methods. Here are the items you can control.

Display or hide the ABC window ([Visible](#)).

Make the ABC window the active window ([Activate](#)).

Change ABC to an icon, maximize the window, and restore it to its previous state ([Minimize](#), [Maximize](#), [Restore](#)).

Arrange minimized chart windows ([Arrangelcons](#)).

Cascade or tile the chart windows ([CascadeCharts](#) and [TileCharts](#)).

Set the position and size of the ABC window ([Top](#), [Left](#), [Right](#), [Bottom](#), [Height](#), and [Width](#)).

Set the title bar text of the ABC window ([Caption](#)).

Set the status bar text of the ABC window ([StatusBar](#)).

Control whether the Field viewer, Notes viewer, Index, and Shape palette are visible in the ABC window ([FieldViewerVisible](#), [NoteViewerVisible](#), [IndexVisible](#), and [ShapePaletteVisible](#)).

Get information about ABC ([FullName](#), [Path](#), [Name](#), [Version](#), and [OperatingSystem](#)).

Set ABC preferences ([SmartShapeSpacing](#), [SSSHorizontal](#), [SSSVertical](#), [ChannelAlignment](#), [TouchAlignment](#), [AlignToRulers](#), [ShowRulers](#), [ShapeSizing](#), [LineSpacingX](#), [LineSpacingY](#), [LinkIndicator](#), [LinkShadow](#), [LaunchIndicator](#), [LaunchShadow](#), [NoteIndicator](#), [NoteShadow](#), [NumberFont](#), [ShowNodesOnLines](#), [FieldPlacement](#), [FieldNamesHidden](#), [FieldsOpaque](#), [FieldFont](#), [FieldsHoursPerDay](#), and [FieldsDaysPerWeek](#)).

Display Windows Help files ([Help](#)).

Undo actions ([UndoAvailable](#) and [Undo](#)).

Close ABC ([Quit](#)).

[Bringing ABC or a Chart to the Front](#)

[Minimizing, Maximizing, and Restoring a Window](#)

[Arranging ABC Icons](#)

[Arranging ABC Charts](#)

[Positioning and Resizing the ABC Window](#)

[Changing the ABC Title Bar](#)

[Changing the ABC Status Bar](#)

[Displaying the Field Viewer, Notes Viewer, Index, and Shape Palette](#)

## Related Topics

### Description

[Running and Setting Up ABC](#)

## Bringing ABC or a Chart to the Front

Related

You bring ABC to the front using the [Activate method](#) of the Application object. You usually have to do this only after the user has done something that moves ABC to the back, such as clicking another application that is visible on the screen or switching to another application using **Alt+Tab** or **Ctrl+Esc**.

ABC.Activate

The [Activate method](#) of the Application object is equivalent to clicking the ABC window to bring it to the front, or using **Alt+Tab** or **Ctrl+Esc** to bring it to the front.

You bring a particular chart to the front using the [Activate method](#) of the Chart object.

Chart.Activate

The [Activate method](#) of the chart object is equivalent to clicking the chart to bring it to the front or choosing the name of the chart from the Window menu.

## Related Topics

### Description

[Controlling the ABC Window](#)

## Minimizing, Maximizing, and Restoring a Window

Related

You can minimize the ABC window and the chart windows to icons using the [Minimize method](#) of the Application object and Chart object. For example, the following statement makes ABC into an icon at the bottom of the Windows screen.

```
ABC.Minimize
```

The following statement makes the chart into an icon at the bottom of the ABC window.

```
Chart.Minimize
```

After you minimize charts to icons, you can arrange them as described in [Arranging ABC Icons](#).

You can use the [Maximize method](#) of the Application object and Chart object to maximize the ABC and chart windows. For example, the following statements maximize ABC and then maximize the chart in it.

```
ABC.Maximize  
Chart.Maximize
```

After you change the size of the ABC window or a chart window, you can change it to its previous size using the [Restore method](#) of Application object and Chart object. For example, the following statements restore the chart and then the ABC windows to their previous sizes.

```
Chart.Restore  
ABC.Restore
```

You can resize the ABC window as described in [Positioning and Resizing the ABC Window](#). You can cascade and tile charts as described in [Arranging ABC Charts](#).

## Related Topics

### Description

[Controlling the ABC Window](#)

## Arranging ABC Icons

Related

When you have several ABC chart windows minimized to icons, you can arrange them at the bottom of the ABC window using the [Arrangelcons method](#) of the Application object.

ABC.ArrangeIcons

The [Arrangelcons method](#) is equivalent to opening the ABC Window menu and choosing Arrange Icons.

## Related Topics

### Description

[Controlling the ABC Window](#)

## Arranging ABC Charts

Related

You can cascade and tile ABC charts when more than one is open. You use the [CascadeCharts method](#) of the Application object to cascade the chart windows and the [TileCharts method](#) of the Application object to tile the chart windows.

```
ABC.CascadeCharts  
ABC.TileCharts
```

These methods are equivalent to opening the ABC Window menu and choosing Cascade or Tile.

## Related Topics

### Description

[Controlling the ABC Window](#)

## Positioning and Resizing the ABC Window

Related

You can specify the position and size of the ABC window at any time. You use the [Top property](#) to set the top edge of the ABC window, the [Left property](#) to set its left edge, the [Right property](#) to set its right edge, and the [Bottom property](#) to set its bottom edge. You can specify the height of the window using the [Height property](#) and the width of the window using the [Width property](#). All these properties are in the Application object. See [Minimizing, Maximizing, and Restoring Windows](#) for more information on controlling the appearance of the ABC window.

When you apply these properties to the ABC window, you specify the position in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high.

You also can specify the height and width of the ABC window. The following example places the ABC window flush with the top and left edges of the screen, and extends it 400 pixels to the right and 500 pixels down from the top.

```
ABC.Top = 0
ABC.Left = 0
ABC.Width = 400
ABC.Height = 500
```

You can use the current values of [Top](#), [Bottom](#), [Left](#), [Right](#), [Height](#), and [Width](#) in statements. The following example places the ABC window in the top half of the screen.

```
Sub Main()
    Dim ABC As Object
    Dim MaxHeight As Integer, MaxWidth As Integer

    Set ABC = CreateObject("ABCFlow.application")

    ABC.Visible = False

    ABC.Maximize           ' Maximize ABC window
    MaxHeight = ABC.Height ' Find height when maximized
    MaxWidth = ABC.Width   ' Find width when maximized
    ABC.Restore

    ABC.Top = -4           ' The -4 is for the window sizing border
    ABC.Left = -4         ' and can be different with other
    ABC.Width = MaxWidth + 4 ' resolutions and Windows setups
    ABC.Height = MaxHeight / 2

    ABC.Visible = True
End Sub
```

## Related Topics

### Description

[Controlling the ABC Window](#)

## Changing the ABC Title Bar

Related

You can customize ABC by changing what it says in the title bar. You set the text of the title bar using the [Caption property](#) of the Application object.

The following statement changes the ABC title bar to *Custom Application* followed by a hyphen and the name of the active chart. ABC appends a hyphen and then the name of the active chart to the caption.

```
ABC.Caption = "Custom Application"
```

You can use the current value of [Caption](#) in statements. For example, the following statements set the text in the title bar to *Second Window* if its value is *First Window*.

```
If ABC.Caption = "First Window" Then  
    ABC.Caption = "Second Window"  
End If
```

## Related Topics

### Description

[Controlling the ABC Window](#)

## Changing the ABC Status Bar Related

You can use the text in the ABC status bar to give hints and feedback to your users. You set the text in the status bar using the [StatusBar property](#) of the Application object.

### Related

The status bar can hold approximately 90 characters, including spaces. The exact number it can hold depends on the characters. (For example, "m" takes up more room than "i" does.)

After running the following statements, if you change the pointer to the selection pointer and double click on a shape, the ABC status bar text changes to "You double clicked!"



**Note:** You must put the ABC Events VBX in the Form window and put the following statement in the subroutine for double clicking.

```
Sub ABC1_DoubleClickSUBCLASS ()
    ABC1.App.StatusBar = "You double clicked!"
End Sub
```

You also must add the statement below in the startup or initialization code of your program. See [Registering Event Procedures](#).

```
ABC.RegisterEvent ABC1.VBX, "Double click example", "DoubleClickSUBCLASS"
```

**Note:** You can restore the normal status bar hints by setting the **StatusBar** property to "".

Enter the following statement after the [StatusBar](#) statement to specify that the action of double clicking will not also cause the normal response, such opening the Link dialog box or opening a linked chart. For more information on **Override**, see [What Are ABC Events?](#)

```
ABC1.Override = True
```

You can use the current value of the [StatusBar property](#) in statements. For example, the following statements add text to the status bar if its value is First Window.

```
If ABC.StatusBar = "First Window" Then
    ABC.StatusBar = "First Window" + " View"
End If
```

## Related Topics

### Description

[Controlling the ABC Window](#)

## Displaying the Field Viewer, Notes Viewer, Index, and Shape Palette

### Related

You can control whether the Field viewer, Notes viewer, Index, and Shape palette are visible in the ABC window. The properties you use are [FieldViewerVisible](#), [NoteViewerVisible](#), [IndexVisible](#), and [ShapePaletteVisible](#). You set them to True or False to show or hide the windows.

Enter the following statement to make the Field viewer visible if fields are defined.

```
ABC.FieldViewerVisible = True
```

Enter the following statement to make the Field viewer invisible.

```
ABC.FieldViewerVisible = False
```

To make the other windows visible or invisible, substitute [NoteViewerVisible](#), [IndexVisible](#), or [ShapePaletteVisible](#) for [FieldViewerVisible](#).

The [FieldViewerVisible property](#) is equivalent to opening the Field menu and choosing Show Viewer or Hide Viewer.



The [NoteViewerVisible property](#) is equivalent to clicking the Note button on the ABC screen.



The [IndexVisible property](#) is equivalent to clicking the Index button on the ABC screen.



The [ShapePaletteVisible property](#) is equivalent to clicking the Shape Palette button on the ABC screen.

You can use the current value of the window properties in statements. For example, the following statements make the Shape palette invisible if the Note viewer is visible.

```
If ABC.NoteViewerVisible = True Then  
    ABC.ShapePaletteVisible = False  
End If
```

## Related Topics

### Description

[Controlling the ABC Window](#)

## Getting ABC System Information

Related

You can get information about the ABC program that is running using the properties [FullName](#), [Path](#), [Name](#), [Version](#), and [OperatingSystem](#) of the Application object.

**Note:** You cannot change the values of the [FullName](#), [Path](#), [Name](#), [Version](#), or [OperatingSystem](#) properties.

The [FullName property](#) returns the fully qualified path of the ABC program that is running, including the name of the executable file.

For example, the following statement puts the fully qualified path of the running ABC in the status bar. The result when you run the program might be a status bar entry of "Full path is C:\WINDOWS\ABC\ABC.EXE."

```
ABC.StatusBar = "Full path is " + ABC.FullName + "."
```

The [Path property](#) returns the fully qualified path of the ABC application that is running, excluding the name of the executable file. For example, the following statement puts the fully qualified path of the running ABC in the status bar. The result when you run the program might be a status bar entry of "Path is C:\WINDOWS\ABC."

```
ABC.StatusBar = "Path is " + ABC.Path + "."
```

The [Name property](#) returns the name of the application running. It always equals "ABC FlowCharter" for compatibility with all ABC products. For example, the following statement puts "ABC FlowCharter" in the status bar.

```
ABC.StatusBar = ABC.Name
```

The [Version property](#) returns the version of the ABC OLE Automation application object that is running. For example, it equals "1.0" if you are running ABC ToolKit version 1.1. The following statement puts the version number in the status bar.

```
ABC.StatusBar = "ABC OLE Automation Version " + ABC.Version + "."
```

The [OperatingSystem property](#) returns a value according to the DOS and Windows version under which ABC is running. For example, it equals "DOS 6.21;Windows 3.11" if you are running those versions. The following statement puts the operating system description in the status bar.

```
ABC.StatusBar = "OS: " + ABC.OperatingSystem + "."
```

## Related Topics

### Description

[Controlling the ABC Window](#)

## Customizing Preferences Related

You can use ABC OLE Automation to set many of the ABC preferences that you can set from the application. The object that contains the preferences is Preferences. You use all the preference properties in the same way. Using the preference properties is described at the end of this section.

**Note:** Some of the options in the Preferences dialog box are not in the Preferences object.

[Alignment Options](#)

[Shape Sizing Options](#)

[Line Options](#)

[Indicator Options](#)

[Field Options](#)

[Setting Preferences](#)

## Related Topics

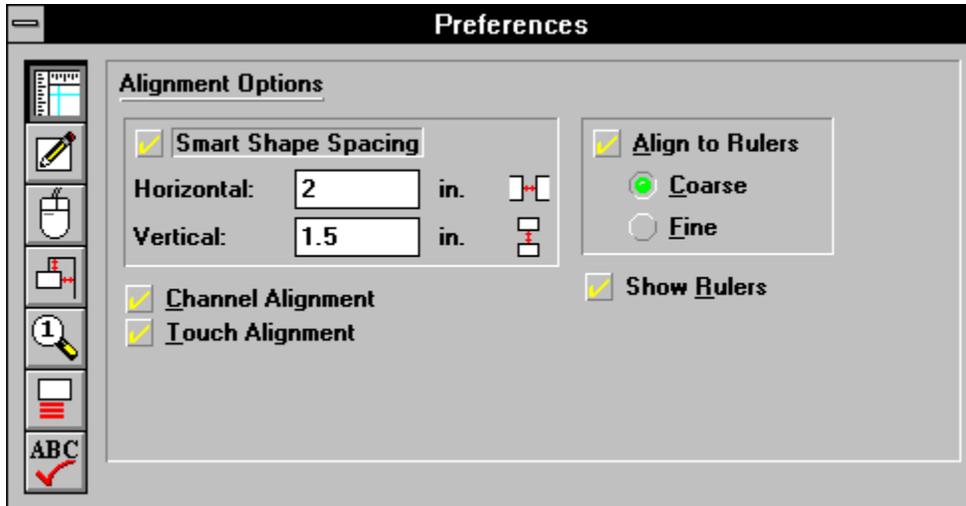
### Description

[Controlling the ABC Window](#)

## Alignment Options

Related

The alignment preference properties are [SmartShapeSpacing](#), [SSSHorizontal](#), [SSSVVertical](#), [ChannelAlignment](#), [TouchAlignment](#), [AlignToRulers](#), and [ShowRulers](#) of the Preferences object.



The following table shows the possible values of the alignment preference properties.

Property	Values
<a href="#">SmartShapeSpacing</a>	True = Selected; False = Not Selected
<a href="#">SSSHorizontal</a>	Number
<a href="#">SSSVVertical</a>	Number
<a href="#">ChannelAlignment</a>	True = Selected; False = Not Selected
<a href="#">TouchAlignment</a>	True = Selected; False = Not Selected
<a href="#">AlignToRulers</a>	0 = Not Selected; 1 = Coarse; 2 = Fine
<a href="#">ShowRulers</a>	True = Selected; False = Not Selected

## Related Topics

### Description

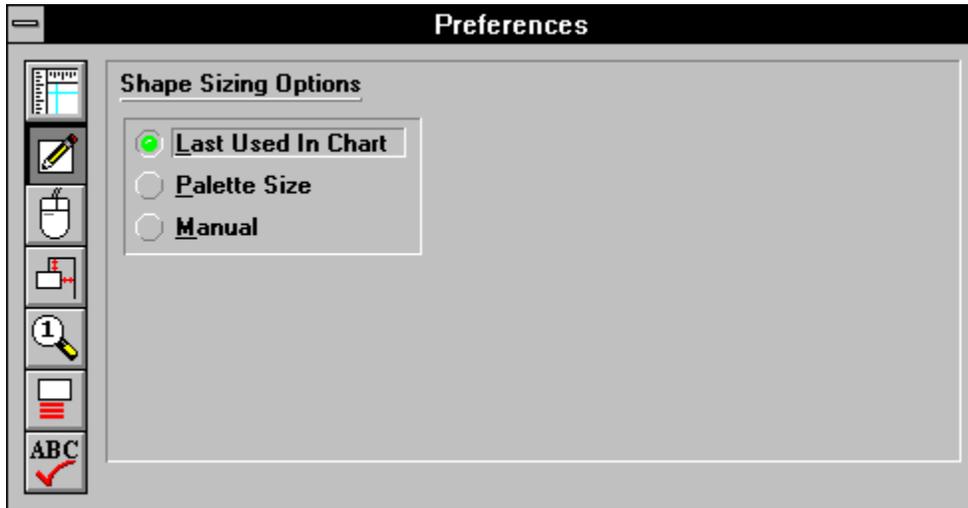
[Customizing Preferences](#)

[Setting Preferences](#)

## Shape Sizing Options

Related

The shape sizing preference property is [ShapeSizing](#) of the Preferences object.



The following table shows the possible values of the shape sizing preference property.

Property	Values
<a href="#">ShapeSizing</a>	0 = Manual; 1 = Last Used; 2 = Palette Size

## Related Topics

### Description

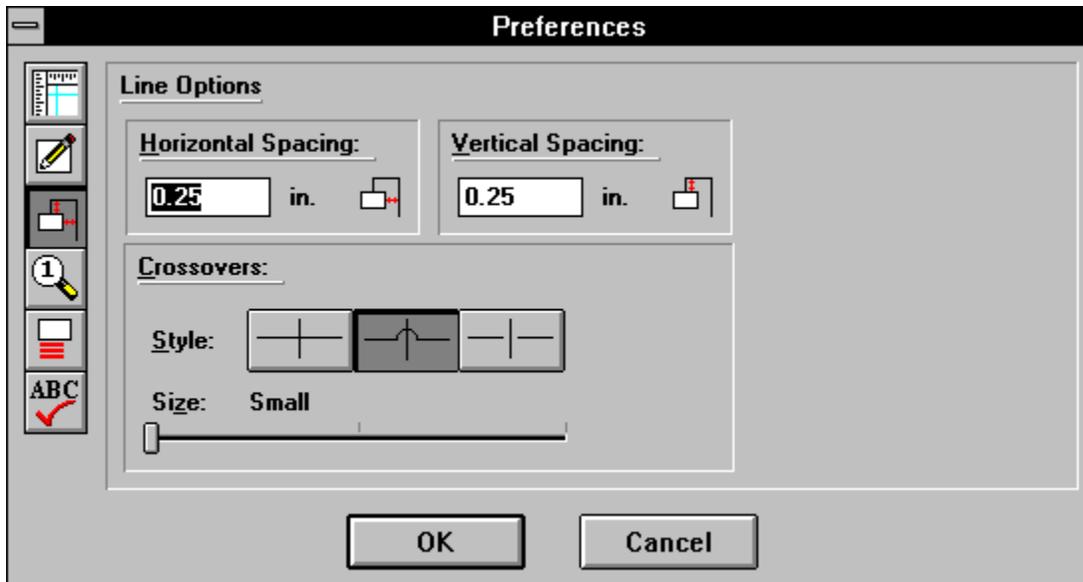
[Customizing Preferences](#)

[Setting Preferences](#)

## Line Options

Related

The line spacing options in the preference properties are [LineSpacingX](#) and [LineSpacingY](#) of the Preferences object, and [LineCrossoverStyle](#) and [LineCrossoverSize](#) of the Chart object.



The following table shows the possible values of the line spacing options preference properties.

Property	Values
<a href="#">LineSpacingX</a>	Value
<a href="#">LineSpacingY</a>	Value

The following table shows the possible values of the [LineCrossoverStyle](#) property.

Style	Description
0	 Bunny hops
1	 Broken lines
2	 Solid lines

The following table shows the possible values of the [LineCrossoverSize](#) property.

RelativeSize	Description
0	 Small
1	 Medium
2	 Large

## Related Topics

### Description

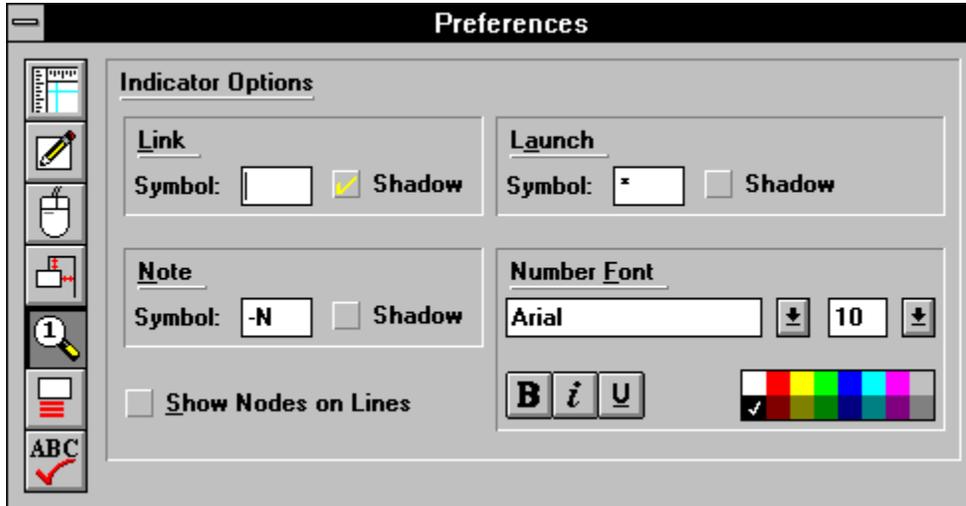
[Customizing Preferences](#)

[Setting Preferences](#)

## Indicator Options

Related

The indicator options are [LinkIndicator](#), [LinkShadow](#), [LaunchIndicator](#), [LaunchShadow](#), [NoteIndicator](#), [NoteShadow](#), [NumberFont](#), and [ShowNodesOnLines](#). These options are part of the Shape object, not the Preferences object.



The following table shows the possible values of the indicator properties.

Property	Values
<a href="#">LinkIndicator</a>	String ("" by default)
<a href="#">LinkShadow</a>	True = Selected; False = Not Selected
<a href="#">LaunchIndicator</a>	String ("*" by default)
<a href="#">LaunchShadow</a>	True = Selected; False = Not Selected
<a href="#">NoteIndicator</a>	String ("-N" by default)
<a href="#">NoteShadow</a>	True = Selected; False = Not Selected
<a href="#">NumberFont</a>	Font object
<a href="#">ShowNodesOnLines</a>	True = Selected; False = Not Selected

## Related Topics

### Description

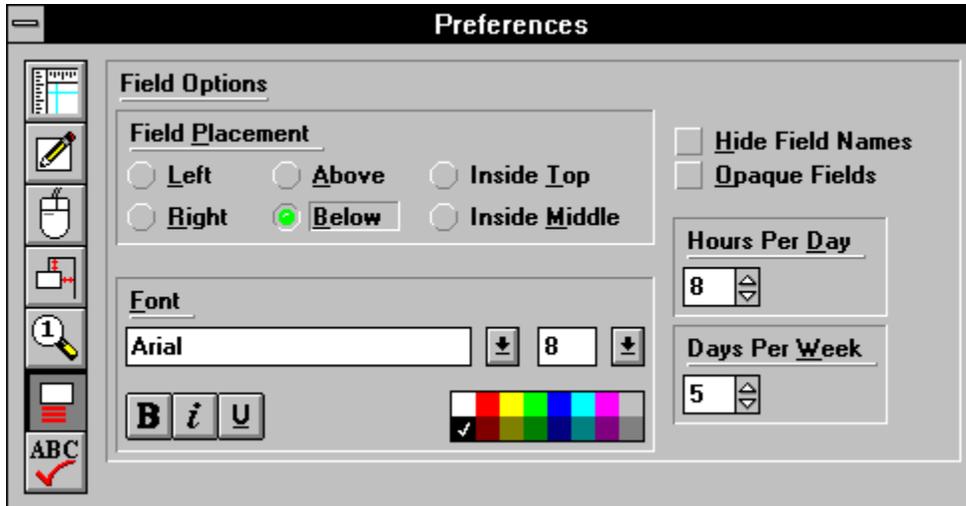
[Customizing Preferences](#)

[Setting Preferences](#)

## Field Options

Related

The field options are [FieldPlacement](#), [FieldNamesHidden](#), [FieldsOpaque](#), [FieldFont](#), [FieldsHoursPerDay](#), and [FieldsDaysPerWeek](#). These options are part of the Chart object, not the Preferences object.



The following table shows the possible values of the indicator properties.

Property	Values
<a href="#">FieldPlacement</a>	0 = Left, 1 = Right, 2 = Above, 3 = Below, 4 = Inside Top, 5 = Inside Middle
<a href="#">FieldFont</a>	Object
<a href="#">FieldNamesHidden</a>	True = Selected; False = Not Selected
<a href="#">FieldsOpaque</a>	True = Selected; False = Not Selected
<a href="#">FieldsHoursPerDay</a>	Number
<a href="#">FieldsDaysPerWeek</a>	Number

For more information on the data field preferences, see [Setting Data Field Preferences](#).

## Related Topics

### Description

[Customizing Preferences](#)

[Setting Preferences](#)

## Setting Preferences

Related

You set all preferences in approximately the same way. You access the information about preferences using the [Preferences property](#) of the Application object. For example, you can set **AlignToRulers** to fine.

```
ABC.Preferences.AlignToRulers = 2
```

To turn aligning to rulers off, set the value to 0. To set the alignment to coarse, set the value to 1. You set the other preferences in the same way by changing **AlignToRulers** to the appropriate property and changing the value to what you want.

You set preferences that are part of the Chart object instead of the Preferences objects slightly differently. For example, you can turn on **FieldNamesHidden**.

```
ABC.Chart.FieldNamesHidden = True
```

You can use the current value of preferences in statements. For example, the following statements put the current setting of the alignment preference in the status bar.

```
If ABC.Preferences.AlignToRulers = 0 Then
    ABC.StatusBar = "Alignment: Off"
ElseIf ABC.Preferences.AlignToRulers = 1 Then
    ABC.StatusBar = "Alignment: Coarse"
Else
    ABC.StatusBar = "Alignment: Fine"
End If
```

You use the other preferences in the same way by changing **AlignToRulers** to the appropriate property and changing the values as appropriate.

## Related Topics

### Description

[Alignment Options](#)

[Customizing Preferences](#)

[Field Options](#)

[Indicator Options](#)

[Line Options](#)

[Shape Sizing Options](#)

## Setting Defaults

**Related**

You can set the defaults for objects that are in the Shape, Line\_, and TextBlock objects. You use the [SetDefaults method](#) of the Chart object. You set the defaults by passing the method an object that has the defaults you want to set.

The following table lists the defaults that you set when you use the **SetDefaults** method.

Object Type	Defaults Set
Shape	Border color, border style, border width, fill color, fill pattern, shadow offset, shadow color, numbers on or off, font properties, text alignment
Line_	Color, width, style, source arrow size, source arrow style, source arrow color, destination arrow size, destination arrow style, destination arrow color
TextBlock	Font properties, text alignment

For example, the following statements first create an object that has shape numbering turned on, has a dark gray fill, and has a dark gray shadow to its lower right. Then the statements set the Shape defaults using that statement. Finally the statements delete the object using the [Clear\\_ method](#) of the Object object.

```
Set DefaultObj = Chart.DrawShape           ' Create an object to have defaults
DefaultObj.NumberShown = True             ' Assign defaults
DefaultObj.FillColor = ABC.GRAY
DefaultObj.ShadowColor = ABC.DK_GRAY
DefaultObj.ShadowStyle = 1
Chart.SetDefaults DefaultObj              ' Set defaults
DefaultObj.Clear_                          ' Delete the object
```

You set the defaults for Line\_ and TextBlock objects similarly.

## Related Topics

### Description

[Speeding Actions](#)

[Running and Setting Up ABC](#)

## Customizing ABC

Related

You can customize ABC in several ways. You can add and remove menus using methods and properties of the Application object, the Menu collection, and the MenuItem object. To make it easy to run other programs, you can add and remove buttons in the toolbox using the methods of the Application object. To provide feedback to the user, you can display messages in dialog boxes, show a percent-complete gauge, and change the pointer to an hourglass using methods and properties of the Application object.

None of these methods and properties have equivalents in ABC.

[Adding Menus](#)

[Adding Buttons](#)

[Providing Feedback](#)

## Related Topics

### Description

[Running and Setting Up ABC](#)

## Adding Menus

Related

You can add as many new menus to ABC as you wish. If you add more than fit on the menu bar, it wraps so all the menus fit. It is best not to have too many menus so the user can easily find the menu he or she wants to use. Each menu should contain commands related to the same type of activity.

You add a menu to ABC using the [AddMenu method](#) of the Application object. The menu is added to ABC at the left of the Window menu, so you set the order of the menus by the order in which you create them. The menu is added to the Menu collection. The [AddMenu method](#) requires three parameters and allows a fourth parameter.

The first parameter is the title of the menu. For example, you might give it the title "&Costs" (the & underlines the next character and makes it the mnemonic).

The second parameter identifies the VBX that receives notification events when the menu is used. Normally you use ABC1.VBX, which registers menus for the [AppMenuSUBCLASS event](#). When the VBX shuts down (when the program ends), the menu is removed from ABC. (See [Handling ABC Events](#) for more information on event handling.)

The third parameter is the name of the program adding the menu to ABC. The easiest way to identify the program is using Form1.Caption.

The fourth parameter, which is optional, lets you specify a chart type for the menu. A chart type is a hidden string field up to eight characters in length indicating the chart type. This field is never used within ABC, but it is useful within an ABC events VBX. For example, if two OLE Automation programs are running, you could change the fourth parameter to avoid conflicts. See [Handling ABC Events](#) for more information.

For example, the following statements create a menu named Costs with a mnemonic of C, register it with the [AppMenuSUBCLASS event](#) by specifying ABC1.VBX, and specify the program using Form1.Caption. The menu object is placed in the variable Menu.

```
Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True

Dim CostMenu As Object

Set CostMenu = ABC.AddMenu("&Costs", ABC1.VBX, Form1.Caption)
```

It is generally best to add any menus that will be necessary at the beginning of a session and leave them until ABC closes. All added menus are automatically removed when ABC closes. Changing menus during a session can be disconcerting to a user.

If you wish, you can change the name of the menu after you create it using the [Text property](#) of the Menu collection. For example, the following statement changes the name of a menu to Cost with a mnemonic of C.

```
CostMenu.Text = "&Cost"
```

You can temporarily make a menu invisible and then show it again using the [Visible property](#) of the Menu collection. For example, the following statement hides a menu.

```
CostMenu.Visible = False
```

If you need to remove a menu while ABC and your program are running, you can use the [RemoveMenu method](#) of the Application object. For example, the following statement removes the menu that was created with the title "&Costs" from ABC and from the Menu collection.

```
ABC.RemoveMenu "&Costs"
```

After you create a menu, you can add items to it. You add items below any existing items using the [AppendItem method](#) and add items in specific places using the [InsertItem method](#), both of the Menu collection.

If you use the name of an existing menu, the methods return the existing MenuItem object. Otherwise they return the new MenuItem object.

With the [AppendItem method](#), you provide the title of the item you wish to create. For example, the following statements create the menu item "Overruns" with a mnemonic of O and place it in the OverrunItem object.

```
Dim OverrunItem As Object
Set OverrunItem = CostMenu.AppendItem("&Overruns")
```

With the [InsertItem method](#), you provide the title of the item you wish to create, as with the [AppendItem method](#), followed by the position of the item. You can specify the position by giving the name of the existing item that the new item should be placed before or by specifying the numerical position of the item. All other items are shifted down. For example, the following statements create the menu item "Explanation" with a mnemonic of E after the existing menu item "&Overruns."

```
Dim ExplanationItem As Object
Set ExplanationItem = CostMenu.InsertItem("&Explanation", "&Overruns")
```

The following statements create the menu item "Overtime" with a mnemonic of V in the second position in the menu.

```
Dim OvertimeItem As Object
Set OvertimeItem = CostMenu.InsertItem("O&vertime", 2)
```

With both methods, you can use the title "-" to create a separator (a solid horizontal line) to divide items into logical groups.

You indicate the status of some types of menu items with a check mark in front of the item. You use the [Checked property](#) of the MenuItem object to set and remove check marks. For example, the following statement puts a check mark to the left of a menu item.

```
CostItem.OvertimeItem.Checked = True
```

When a menu item is not available because choosing it is inappropriate in the current situation, you make the item gray. You use the [Enabled property](#) of the MenuItem object to gray a menu item. The following statement turns an item gray.

```
OvertimeItem.Enabled = False
```

If you wish, you can change the name of a menu item after you create it using the [Text property](#) of the MenuItem object. For example, the following statement changes the name of a menu item to Overtime with a mnemonic of T.

```
OvertimeItem.Text = "Over&time"
```

You can remove an item from a menu using the [DeleteItem method](#) of the Menu collection. For example, the following statement removes an item from a menu.

```
CostMenu.DeleteItem OvertimeItem
```

If you wish, you can delete all the items from a menu using the [DeleteAll method](#) of the Menu collection.

```
CostMenu.DeleteAll
```

You can find the menu item that is in a specific position using the [Item method](#) of the Menu collection. You specify either the position of the item in the menu or the text in the item. For example, the second statement puts into CurrentItem the Menu object that is in the third position in the menu. The third statement puts into NextItem the Menu object that has the text Over&time.

```
Dim CurrentItem As Object, NextItem As Object  
Set CurrentItem = CostMenu.Item (3)  
Set NextItem = CostMenu.Item("Over&time")
```

## Related Topics

### Description

[Customizing ABC](#)

## Adding Buttons

Related

You can add up to ten buttons to the toolbox or toolbar in ABC. These buttons let you quickly run other programs, both programs you have written and commercial programs, such as Microsoft Excel.

You create buttons using the [CreateAddOn method](#) of the Application object. The first parameter is the position of the button. It can be 1 through 10, or you can use -1 for the first available position. **Note:** In ABC Viewer, you can only use the numbers 3 through 10 because numbers 1 and 2 are for the DataAnalyzer button and SnapGraphics button, which are not in ABC Viewer.

The second parameter is the name of the button. The text you enter is used in bubble help and in the hint line. For example, if you enter a second parameter of "Excel" (with no punctuation), the bubble help is "Run Excel." (with "Run" before it and a period after it). The hint line is "Click to run Excel." (with "Click to" before it and a period after it).

The third parameter is the name of the program to run, including the fully qualified path.

The fourth parameter, which is optional, is the location of the icon you want to use for the button, including the fully qualified path. The button is 25 X 25 pixels or 17 X 17 pixels (depending on the version of ABC you are using), so standard icons (which are 32 X 32 pixels) are shrunk to fit. Alternatively, you can provide a BMP file to use as the button. If the BMP file is smaller than required, the bitmap is centered. If you do not provide this parameter, the icon in the program to run is used.

A fifth parameter, not used in the current version, is flags reserved for future use.

For example, the following statement adds the button for the Deployment Wizard in the third position in the toolbox.

```
ABC.CreateAddOn 3, "Deployment Wizard", "c:\windows\system\deploy.exe"
```

When you use the [CreateAddOn method](#), a button is added. The information describing the button is also added to the ABC.INI file in the [ABCFlowCharter] section. The first button location is AddOn1, the second is AddOn2, and so on. For example, the following ABC.INI entries create buttons in positions 1 and 2.

```
AddOn1="Excel","C:\WINDOWS\EXCEL\EXCEL.EXE"  
AddOn2="Deployment Wizard","C:\WINDOWS\SYSTEM\DEPLOY.EXE"
```

The buttons you create using the [CreateAddOn method](#) appear each time ABC is run until you remove them using the [RemoveAddOn method](#) of the Application object.

With the [RemoveAddOn method](#), you specify either the position of the button (1 though 10, but 3 through 10 in ABC Viewer) or the name of the program to run, including the fully qualified path. For example, the first line below removes the third button, and the second line removes the button for Excel regardless of its position in the toolbox.

```
ABC.RemoveAddOn 3  
ABC.RemoveAddOn "C:\WINDOWS\EXCEL\EXCEL.EXE"
```

## Related Topics

### Description

[Customizing ABC](#)

## Providing Feedback

**Related**

You can provide feedback to your users in several ways, from as simple a thing as changing the cursor to indicate that the user should wait for a moment to more complex things such as posting a gauge that shows how far an operation has progressed, showing a hint line, and posting a dialog box that the user must respond to.

When you have an operation that will take a long time (anything approaching a second) and the user cannot usefully click somewhere (such as on Cancel), it is customary to change the cursor. You can change the cursor to the wait cursor using the [Hourglass property](#) of the Application object.

```
ABC.Hourglass = True
```

Posting a gauge that shows the progress of an operation uses the [PercentGauge method](#), the [PercentGaugeValue property](#), the [HidePercentGauge method](#), and the [PercentGaugeCancelled method](#), all of the Application object.

You create a percent gauge, with its value set to 0, using the [PercentGauge method](#). It takes three optional parameters. The first is the name that goes in the title bar. The second is the first line of text above the gauge. The third is the second line of text above the title bar.

For example, the following statement creates the gauge shown.

```
ABC.PercentGauge "Object Creation", "Creating objects.", "Click Cancel to stop."
```



After you create a gauge, you set its value using the [PercentGaugeValue property](#). Set it equal to a number from 0 to 100 to have the gauge show the appropriate position. For example, if the operation is 53% complete, the following statement makes the gauge show that value.

```
ABC.PercentGaugeValue = 53
```

You check to see if the user has chosen the Cancel button in the gauge using the [PercentGaugeCancelled method](#). For example, the following statement sets the value of CancelCreation to True or False depending on whether the user has chosen the Cancel button.

```
CancelCreation = ABC.PercentGaugeCancelled
```

After the operation is complete, or if the user clicks Cancel, you need to remove the gauge. You do that using the [HidePercentGauge method](#).

```
ABC.HidePercentGauge
```

Most often, the value for the [PercentGaugeValue property](#) is a variable that you compute immediately before changing the value of the gauge. For example, the following statements determine the completion value and change the gauge only if the percentage is different. To avoid slowing the loop, it only redraws the gauge when the percentage has changed by at least 1. This example also shows the use of the

## PercentGaugeCancelled method and the HidePercentGauge method.

```
ABC.PercentGauge "Object Creation", "Creating objects.", "Click Cancel to stop."

OldPercentDone = 0
CreateCount = 1000

For Creation = 1 to CreateCount

    CancelCreation = ABC.PercentGaugeCancelled           ' Cancelled?
    If CancelCreation Then
        ABC.HidePercentGauge                             ' Get rid of gauge
        Exit For                                         ' Leave creation loop
    End If

    Chart.DrawShape                                     ' Create the shape

    PercentDone = Int(Creation / CreateCount * 100)     ' Find percentage done
    If PercentDone <> OldPercentDone Then               ' Has percentage changed?
        ABC.PercentGaugeValue = PercentDone             ' Set gauge
        OldPercentDone = PercentDone                   ' Reset comparison value
    End If

Next Creation

ABC.HidePercentGauge
```

You can show a hint line using the Hint method of the Application object. You usually use a hint line to describe a menu, command, or button you have added, or to give information about a percentage gauge or dialog box you have posted. You most often use the events [AppMenuHintSUBCLASS](#), [AppMenuSUBCLASS](#), and [AppMenuPopupSUBCLASS](#). (See [Handling ABC Events](#) for more information on events.) For example, the following line puts the line "Click Cancel to stop creation." in the hint line.

```
ABC.Hint "Click Cancel to stop creation."
```

Note that the hint line you place only stays until the user moves the mouse so that a different hint line appears. If you want to make the hint line stay until you change it to "" you should use the [StatusBar method](#).

You can post a dialog box using the [MsgBox method](#) of the Application object. The method is similar to the MsgBox function used in the Basic programming language, with three parameters. The first parameter is the message that goes in the dialog box.

The second parameter, which is optional, defines the type of dialog box. If you omit the second parameter, the value is 0.

The third parameter, which is optional, sets the title bar text of the dialog box. If you omit the third parameter, the title of the dialog box is "Micrografx ABC FlowCharter."

The value of the second parameter can be the sum of values from the table. For example, if you want to show a Stop icon along with Yes and No buttons, the value of the second parameter is 19. The easiest way to set the values is to use the constants and sum them into a variable. The following table shows the values available. **Note:** In Visual Basic, these values have constants associated with them, such as MB\_OK. Those constants are not available for ABC OLE Automation.

Value	Effect
0	Display the OK button only.
1	Display the OK and Cancel buttons.
2	Display the Abort, Retry, and Ignore buttons.
3	Display the Yes, No, and Cancel buttons.
4	Display the Yes and No buttons.



## Related Topics

### Description

[Customizing ABC](#)

## Displaying Help

**Related**

You can display help at any time that you wish, based on the actions of your user. You use the **Help** method of the Application object to display the help topic of your choice.

There are two optional parameters you can use with the [Help method](#). If you use the [Help method](#) with no parameters, the ABC help appears showing the default topic.

The first parameter, a text string, specifies a Windows help file. You can use it to specify a help file other than the one shipped with ABC, so that you can direct your users to a help file that you prepared for your particular application.

The second parameter, a long or a text string, is a context ID or help context string to call a particular topic in the help file.

The following statement opens the Help window with the Shape Tool topic displayed because its context ID is 71681.

```
ABC.Help, 71681
```

If you write your own help, then the statement is something like this.

```
ABC.Help "C:\ABC\MYHELP.HLP", "Getting_Started"
```

Using the [Help method](#) is equivalent to positioning the pointer or opening a dialog box, and then pressing **F1**.

## Related Topics

### Description

[Running and Setting Up ABC](#)

## Closing ABC

Related

When you wish, you can close ABC using the [Quit method](#) of the Application object.

When you use the [Quit method](#), ABC closes. It does not prompt the user to save changes on open files. Before you close ABC, you should save the files you want to be saved as described in [Working with Chart Files](#).

The following statement closes ABC.

```
ABC.Quit
```

Using the [Quit method](#) is equivalent to opening the File menu and choosing Exit, or using any of the other methods to close ABC, except that it does not prompt for saving changes.

## Related Topics

### Description

[Running and Setting Up ABC](#)

## Window Handles

Related

Using window handles is an advanced feature of ABC OLE automation. Window handles are useful for calling the Windows API calls directly from ABC OLE Automation. For more information, open the Visual Basic 3.0 Help and search for "hWnd." Also see [Handling ABC Events](#).

You can get the handle of the windows within ABC, including the handle for the main window, the Field viewer, the Note viewer, the Index, and the Shape palette. The properties associated with the windows are [WindowHandle](#) (of the Application object for ABC and of the Chart object for a specific chart), [FieldViewerWindowHandle](#), [NoteViewerWindowHandle](#), [ShapePaletteWindowHandle](#), and [IndexWindowHandle](#) of the Application object. These properties provide the handle to each of the windows. If the window is not visible, they are Null.

## Related Topics

### Description

[Running and Setting Up ABC](#)

Related

## Handling ABC Events

Related

[What Are ABC Events?](#)

[Registering Event Procedures](#)

[Registering Events and Multiple ABC Applications](#)

[Event Variables](#)

[When ABC Closes](#)

[When Add-On Menus Open](#)

[When MenuItems Are Highlighted](#)

[When MenuItems Are Chosen](#)

[When Charts Open](#)

[When Linked Charts Open](#)

[When New Charts Are Created](#)

[When Charts Are Activated](#)

[When Charts Change](#)

[When Charts Are Pasted](#)

[When Charts Close](#)

[When Objects Are Clicked](#)

[When Objects Are Selected](#)

[When Objects Move](#)

[When Objects Are Resized](#)

[When Objects Are Deleted](#)

[When Shapes Are Double Clicked](#)

[When Shapes Are Drawn](#)

[When Shapes Are Replaced](#)

[When Lines Are Drawn](#)

[When Lines Attach](#)

[When Fonts Change](#)

[When Field Values Change](#)

[When Special Keys Are Pressed](#)

[When Text Changes](#)

## Related Topics

Description

[Contents](#)

## What Are ABC Events?

Related

ABC events are the key to running an automation program. For example, suppose you want an program to turn any shape gray when the user tries to delete it. Since you don't know when the user will delete a shape, and you don't know what shape the user will delete, you want your program alerted each time the user presses the **Del** key.

Pressing the **Del** key is an event. Each time an event occurs, your program can run a procedure specific to that event, known as an *event procedure*. When a user presses the **Del** key, for example, your program can automatically run the [DeleteSUBCLASS event](#) procedure.

Other examples of ABC events are creating a new chart, selecting a shape, moving a shape, replacing a shape, and drawing a line. ABC events can be triggered by actions of the user or by program instructions.

The names of the events recognized by ABC OLE Automation are listed below.

AppQuitNOTIFY	FieldValueChangedNOTIFY
AppQuitSUBCLASS	LinkNOTIFY
AppMenuHintSUBCLASS	NewLineNOTIFY
AppMenuPopupSUBCLASS	NewShapeNOTIFY
AppMenuSUBCLASS	ObjectClickSUBCLASS
ChartActivateNOTIFY	ObjectFontChangeNOTIFY
ChartChangeNOTIFY	ObjectLineAttachNOTIFY
ChartCloseSUBCLASS	ObjectMovedNOTIFY
ChartNewNOTIFY	ObjectMoveSUBCLASS
ChartOpenNOTIFY	ObjectSizedNOTIFY
ChartPasteNOTIFY	ObjectSizeSUBCLASS
DeleteSUBCLASS	ObjectTextChangedNOTIFY
DoubleClickSUBCLASS	ReplaceShapeNOTIFY
ExclusiveSelectionNOTIFY	SpecialKeySUBCLASS

These events provide the capabilities you need to write programs that can interact with the user to create and manipulate ABC charts.

**Warning:** You should not put a method inside an event if that method generates the event. For example, you should not put a call to the **Link** method inside the **LinkNOTIFY** event. If you do so, the program will go into an endless loop, ABC will crash, Visual Basic may crash, and Windows will become unstable.

To tell your program to perform a particular task when an event occurs, you add program instructions to the event procedure for that event. When ABC OLE Automation detects the event, it executes the instructions in the associated event procedure. At the conclusion of the event procedure, ABC OLE Automation returns to a state in which it waits for another event.

**Note:** ABC OLE Automation requires that you identify the event procedures you want executed by registering the events. For more information on registering events, see [Registering Event Procedures](#).

### SUBCLASS and NOTIFY Events

ABC has standard actions that it performs when an event is triggered. These actions, called *standard behaviors*, are independent of any tasks performed by program instructions in the event procedure for a triggered event. For example, the standard behavior for the [AppQuitSUBCLASS event](#) is to close ABC.

ABC events are in two categories, depending on whether the event procedure is executed before or after the event's standard behavior.

- Subclass event procedures execute *before* ABC performs its standard behavior. Subclass event procedures are identified by the use of SUBCLASS in the event's name.
- Notify event procedures execute *after* ABC performs its standard behavior. Notify event procedures are identified by the use of NOTIFY in the event's name.

You can cancel the standard behavior of a subclass event by setting the **Override** property of the ABC1 object to True in the event procedure for the event. *You cannot override the standard behavior of a notify event, because the standard behavior is performed before the event procedure executes.*

The sample below demonstrates overriding the standard behavior of a subclass event.

```
Sub ABC1_DeleteSUBCLASS ( )  
    Dim ABCObj As Object           ' Declare local object variable  
    Set ABCObj = ABC1.Object       ' Set ABCObj to event object  
    Set ABCObj.Text = "Deleted"    ' Set Text of deleted object  
    ABC1.Override = True           ' Override standard behavior  
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## Registering Event Procedures

Related

The purpose of registering events is to identify the event procedures that you want ABC OLE Automation to execute. Registering an event does not determine whether the event executes when triggered, but only whether the procedure associated with the event is executed also.

ABC OLE Automation requires that all event procedures except [AppMenuSUBCLASS](#) be registered before they can be executed. The [AppMenuSUBCLASS event](#) is registered automatically when an add-on menu is created with the [AddMenu method](#) of the Application object.

Normally, you register event procedures in the startup or initialization code of your program. If you want to "turn off" an event procedure that you have registered previously, you can unregister it. When your program ends, all events for all ABC VBXs on any form in your program unregistered are automatically.

You use the [RegisterEvent method](#) of the Application object to register an event procedure. The general syntax for the [RegisterEvent method](#) is shown below.

```
ABC.RegisterEvent VBXName.VBX, IdString, "EventName" [,ChartType]
```

The *VBXName.VBX* parameter identifies the ABC OLE Automation control to which the registered events apply. Unless you have changed the ABC OLE Automation control's Name property from its default setting, *VBXName* is ABC1.

The *IdString* parameter identifies the Visual Basic form on which the ABC OLE Automation control is located. *IdString* is normally the [Caption property](#) setting of the form (Form1.Caption, by default).

The *EventName* parameter is the name of the event being registered. This name must be enclosed in quotes.

The *ChartType* parameter is optional. If *ChartType* is omitted, the registered events apply to all charts. If you wish to register the event for only a particular type of chart, then specify that chart **Type** for this parameter. You set a chart's type with the [Type property](#) of the Chart object.

**Note:** ABC OLE Automation does not permit two or more ABC applications that are running at the same time to register the same events, unless the events are registered for different chart types. For information on this restriction, see [Registering Events and Multiple ABC Applications](#).

Some examples of registering events are shown below. The first example registers the [ChartNewNOTIFY event](#) for all charts in the application. The second example registers the [DeleteSUBCLASS event](#) for charts of the type Hourly.

```
ABC.RegisterEvent ABC1.VBX, "AutoParts", "ChartNewNOTIFY"  
ABC.RegisterEvent ABC1.VBX, "Personnel", "DeleteSUBCLASS", "Hourly"
```

You use the [UnRegisterEvent method](#) of the Application object to unregister an event. The general syntax for the [UnRegisterEvent method](#) is shown below.

```
ABC.UnRegisterEvent VBXName.VBX, "EventName" [, ChartType]
```

*VBXName.VBX* is the name of the ABC OLE Automation control, *EventName* is the name of the event being unregistered, and *ChartType* is the chart **Type**. *EventName* must be enclosed in quotes. *ChartType* is optional.

Some examples of unregistering events are shown below. The first example unregisters the [ChartNewNOTIFY event](#) for all charts in the application. The second example unregisters the [DeleteSUBCLASS event](#) for charts of the type Hourly.

```
ABC.UnRegisterEvent ABC1.VBX, "ChartNewNOTIFY"  
ABC.UnRegisterEvent ABC1.VBX, "DeleteSUBCLASS", "Hourly"
```

## Related Topics

### Description

[Handling ABC Events](#)

## Registering Events and Multiple ABC Applications

Related

ABC OLE Automation imposes a restriction on the registration of events for multiple ABC applications. The same events cannot be registered by two or more ABC applications running at the same time, unless the events are registered for different chart types.

The [AppQuitSUBCLASS](#), [AppQuitSUBCLASS event](#), and [SpecialKeySUBCLASS](#) events are exempt from this limitation and can be registered by multiple ABC applications without restriction.

An example will make this restriction clear. Assume that you are running an ABC application that registers the [DeleteSUBCLASS event](#) using the statement shown below.

```
ABC.RegisterEvent ABC1.VBX, AutoParts, "DeleteSUBCLASS"
```

If you now attempt to run a second ABC application that registers the [DeleteSUBCLASS event](#), you will be notified of the event conflict by ABC and asked to close the first application before the second application can run.

To avoid this kind of event conflict, design your programs so that they deal with specific chart types. This lets you to register events only for those chart types, thereby avoiding any event conflicts with other concurrently running ABC applications.

## Related Topics

### Description

[Handling ABC Events](#)

## Event Variables

Related

ABC OLE Automation passes information to its event procedures by setting various ABC OLE Automation object variables. These object variables are local to the event procedure and are reset each time the event procedure is called. If you need to save the value of one of these ABC OLE Automation object variables between executions of the event procedure, you must save the value of the ABC OLE Automation object variable in a global object variable.

Not all ABC OLE Automation object variables apply to all events. The event definitions that appear in later sections of this chapter describe which ABC OLE Automation object variables are valid for each event.

The ABC OLE Automation variables passed to events are defined below.

<b>Variable</b>	<b>Definition</b>
<i>App</i>	The Application object that triggered this event.
<i>Chart</i>	The Chart object in which this event occurred.
<i>Object</i>	The Object object to which the event applies. This variable is set only if the event applies to a single Object.
<i>Object2</i>	The second object to which the event applies. This variable is only set using the <a href="#">ObjectLineAttachNOTIFY event</a> .
<i>FieldValue</i>	The FieldValue object to which the event applies.
<i>Menu</i>	The Menu object to which this event applies.
<i>MenuItem</i>	The MenuItem object to which this event applies.
<i>WParam</i>	Set only for the <a href="#">SpecialKeySUBCLASS event</a> . (This is a Long variable, not an Object variable.)
<i>LParam</i>	Reserved for future use.
<i>Override</i>	A property that lets you cancel normal ABC behavior in response to an event. It is automatically reset to False at the end of every event call.
<i>VBX</i>	A property that is used for registering events and adding menus to let ABC a communication path to your program.

The following example of an event procedure uses an ABC OLE Automation object variable. It tests the [Type property](#) of the Chart object passed to the event procedure. If the [Type property](#) is PartTime, then the event procedure sets **Override** to True, which cancels the standard behavior for the event.

```
Sub ABC1_ChartMoveSUBCLASS ( )
    If ABC1.Chart.Type = PartTime Then
        ABC1.Override = True
    End If
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)



## Related Topics

### Description

[Handling ABC Events](#)

## When Add-On Menus Open

Related

The [AppMenuPopupSUBCLASS event](#) occurs when the user opens an add-on menu by clicking the menu's name. Add-on menus are created with the [AddMenu method](#) of the Application object. The [AppMenuPopupSUBCLASS event](#) procedure is triggered before ABC displays the add-on menu. The menu that is about to open is passed to the event procedure in the Menu object variable.

Because the [AppMenuPopupSUBCLASS event](#) is triggered before the add-on menu opens, you can use this event procedure to determine whether any items on the add-on menu should be disabled (gray) or checked. A menu item is disabled by setting the [Enabled property](#) of the MenuItem object to False. A menu item is checked by setting the [Checked property](#) of the MenuItem object to True.

An [AppMenuPopupSUBCLASS](#) example is shown below. This example assumes that other code has created an add-on menu with the items Shape Count and Line Count.

```
Sub ABC1_AppMenuPopupSUBCLASS ( )
    Dim ShapeCmd As Object           ' Declare object variable
    Dim LineCmd As Object           ' Declare object variable
    If CurrentShapeCount = 0 Then
        Set ShapeCmd = ABC1.Menu.Item("Shape Count") ' Get MenuItem object
        ShapeCmd.Enabled = False           ' Gray item
    End If
    If CurrentLineCount = 0 Then
        Set LineCmd = ABC1.Menu.Item("Line Count") ' Get MenuItem object
        LineCmd.Enabled = False           ' Gray item
    End If
End Sub
```

The procedure tests the status of the CurrentShapeCount and CurrentLineCount variables to determine whether the Shape Count and Line Count menu items should be disabled. The Shape Count menu item is disabled when the CurrentShapeCount is zero. The Line Count menu item is disabled when the CurrentLineCount is zero.

## Related Topics

### Description

[Handling ABC Events](#)

## When MenuItems Are Highlighted

Related

The [AppMenuHintSUBCLASS event](#) occurs when the user moves the menu cursor to an item on an add-on menu. The [AppMenuHintSUBCLASS event](#) procedure is triggered before ABC highlights the menu item. The menu item to be highlighted is passed to the event procedure in the MenuItem object variable.

An [AppMenuHintSUBCLASS](#) example is shown below. This example illustrates how to use the [AppMenuHintSUBCLASS event](#) procedure to display hint line messages describing the purpose of items on an add-on menu.

```
Sub ABC1_AppMenuHintSUBCLASS ( )
  If ABC1.MenuItem = "Shape Count" Then
    ABC1.App.Hint("Click to display shape count")      ' Show hint
  End If
  If ABC1.MenuItem = "Line Count" Then
    ABC1.App.Hint("Click to display line count")      ' Show hint
  End If
End Sub
```

This example determines which menu item is to be highlighted and displays the appropriate hint line message.

## Related Topics

### Description

[Handling ABC Events](#)

## When MenuItems Are Chosen

Related

The [AppMenuSUBCLASS event](#) occurs when the user chooses an item on an add-on menu. The menu item object that was chosen is passed to the event procedure in the MenuItem variable.

**Note:** Do not register the [AppMenuSUBCLASS event](#). The [AppMenuSUBCLASS event](#) is registered automatically when an add-on menu is created.

An [AppMenuSUBCLASS](#) example is shown below. This example determines which menu item is chosen and executes the general procedure that performs the function of the item. Because it may take some time to count the shapes or lines, the procedure displays the hourglass cursor using the [Hourglass property](#) of Application object before it performs the counting operation. When the counting is complete, it clears the hourglass.

```
Sub ABC1_AppMenuSUBCLASS ( )
    ABC.Hourglass = True                ' Display hourglass
    If ABC1.MenuItem = "Shape Count" Then
        X = Shapes.Count
    End If
    If ABC1.MenuItem = "Line Count" Then
        Y = Object.Count
    End If
    ABC.Hourglass = False              ' Clear hourglass
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Charts Open

Related

The [ChartOpenNOTIFY event](#) occurs when the user opens a new chart file by choosing Open in the File menu of ABC. The [ChartOpenNOTIFY event](#) procedure is triggered following the successful opening of the chart file. The opened chart object is passed to the event procedure in the Chart object variable.

A [ChartOpenNOTIFY](#) example is shown below. This example stores the path and the page count of the newly opened chart in the global variables CurrentPath and CurrentPages.

```
Sub ABC1_ChartOpenNOTIFY ( )
    CurrentPath = ABC1.Chart.FullName           ' Save path of chart
    CurrentPages = ABC1.Chart.PageCount        ' Save chart page count
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Linked Charts Open Related

The [LinkNOTIFY event](#) occurs when a chart file is opened by double-clicking the object to which it is linked. The [LinkNOTIFY event](#) procedure is triggered following the successful opening of the chart file.

The chart object from which the linked chart was opened (the source chart) is passed to the event procedure in the Chart object variable. The linked chart object (the chart just opened) can be obtained using the [ActiveChart property](#) of the Application object. The Object that was double-clicked in the source chart to open the linked chart is passed to the event procedure in the Object object variable.

The [LinkNOTIFY](#) example below saves the source chart, source object, and linked chart in the global object variables SourceChart, SourceObject, and CurrentChart.

```
Sub ABC1_LinkNOTIFY ( )
    SourceChart = ABC1.Chart           ' Save source chart
    SourceObject = ABC1.Object         ' Save source object
    CurrentChart = ABC1.ActiveChart    ' Save linked chart
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When New Charts Are Created

Related

The [ChartNewNOTIFY event](#) occurs when the user creates a new chart by choosing New in the File menu of ABC. The [ChartNewNOTIFY event](#) procedure is triggered following the creation of the new chart. The new chart object is passed to the event procedure in the Chart object variable.

A [ChartNewNOTIFY](#) example is shown below. It enlarges the new chart to full size.

```
Sub ABC1_ChartNewNOTIFY ( )
    ABC1.Chart.Maximize
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Charts Are Activated

Related

The [ChartActivateNOTIFY event](#) occurs when a chart is activated by clicking it or choosing it from a menu. The [ChartActivateNOTIFY event](#) procedure is triggered following the activation of the chart. The activated chart object is passed to the event procedure in the Chart object variable.

The [ChartActivateNOTIFY](#) example shown below tests the [Type property](#) of the activated chart and executes the general procedure DeployEditMode if it is a CHARTTYPE type. If the activated chart is not a CHARTTYPE type, then its [Caption](#) (title) property is set to the standard caption "Micrografx ABC FlowCharter."

```
Sub ABC1_ChartActivateNOTIFY ( )
  If ABC1.Chart.Type = CHARTTYPE Then
    Call DeployEditMode
  Else
    ABC1.App.Caption = ""           ' Set to standard caption
  End If
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Charts Change

Related

The [ChartChangeNOTIFY event](#) occurs when a chart is changed in any way. The [ChartChangeNOTIFY event](#) procedure is triggered following the changing of the chart. The changed chart object is passed to the event procedure in the Chart object variable.

The [ChartChangeNOTIFY](#) example shown below sets the chart's Type property to MODIFIED to indicate that it has been changed.

```
Sub ABC1_ChartChangeNOTIFY ( )  
    ABC1.Chart.Type = MODIFIED  
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Charts Are Pasted Related

The [ChartPasteNOTIFY event](#) occurs when a user pastes something into a chart by pressing the keyboard shortcut **Ctrl+V** or choosing Paste in the Edit menu of ABC. The [ChartPasteNOTIFY event](#) procedure is triggered following the paste. The chart object is passed to the event procedure in the Chart object variable.

A [ChartPasteNOTIFY](#) example is shown below. After a paste operation, the objects pasted into a chart are the only selected objects. This example uses this feature to color the pasted objects blue.

```
Sub ABC1_ChartPasteNOTIFY ( )
  Dim Obj As Object, Objs As Object      ' Declare variables
  ABC.Hourglass = True                  ' Display hourglass

  Set Objs = ABC1.Chart.Objects
  Do
    Set Obj = Objs.ItemFromSelection     ' Get selected object
    Obj.Color = ABC.BLUE                 ' Color it blue
  Loop While Obj                        ' Loop until done

  ABC.Hourglass = False                 ' Clear hourglass
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Charts Close

Related

The [ChartCloseSUBCLASS event](#) occurs when the user closes a chart by choosing Close in the File menu of ABC. The [ChartCloseSUBCLASS event](#) procedure is triggered immediately before the user is prompted to save changes and the chart is closed. The closing chart object is passed to the event procedure in the Chart object variable.

The [ChartCloseSUBCLASS](#) example below calls a general procedure to update an external database when a chart is closed.

```
Sub ABC1_ChartCloseSUBCLASS ( )  
    UpdateDatabase  
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Objects Are Clicked

Related

The [ObjectClickSUBCLASS event](#) occurs when the user clicks an object. The [ObjectClickSUBCLASS event](#) procedure is triggered before ABC shows the Object as selected.

The clicked Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An [ObjectClickSUBCLASS](#) example is shown below. This example tests the [Type property](#) of the active chart and sets the clicked object to Green if it is a PartTime type and to Yellow otherwise.

```
Sub ABC1_ObjectClickSUBCLASS ( )
  If ABC1.Chart.Type = PartTime Then
    ABC1.Object.Color = ABC1.App.GREEN
  Else
    ABC1.Object.Color = ABC1.App.YELLOW
  End If
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Objects Are Selected

Related

The [ExclusiveSelectionNOTIFY event](#) occurs when the user selects a single Object object. The [ExclusiveSelectionNOTIFY event](#) procedure is triggered after ABC shows the Object as selected.

The selected Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An [ExclusiveSelectionNOTIFY](#) example is shown below. It copies any single object selected to the Clipboard.

```
Sub ABC1_ExclusiveSelectionNOTIFY ( )
  If ABC1.Chart.Copy
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Objects Move

**Related**

ABC OLE Automation has two events relating to moving objects: [ObjectMoveSUBCLASS](#) and [ObjectMovedNOTIFY](#).

The [ObjectMoveSUBCLASS event](#) occurs when the user starts to move an Object object. The [ObjectMoveSUBCLASS event](#) procedure is triggered before ABC initiates any move behavior.

The Object about to move is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An [ObjectMoveSUBCLASS](#) example is shown below. This example saves the left and top locations of the object before it is moved in the global variables GLeft and GTop.

```
Sub ABC1_ObjectMoveSUBCLASS ( )
    GLeft = ABC1.Object.Left      ' Save left edge
    GTop = ABC1.Object.Top       ' Save top edge
End Sub
```

The [ObjectMovedNOTIFY event](#) occurs when an Object object is moved. The [ObjectMovedNOTIFY event](#) procedure is triggered after ABC has moved the Object.

The Object that was moved is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An [ObjectMovedNOTIFY](#) example is shown below. This example uses the [Type property](#) of the moved Object to decide its action. For a PHASE type, the procedure sets the top edge of the moved Object to GTop. For a DEPT type, the procedure sets the left edge of the moved Object to GLeft.

```
Sub ABC1_ObjectMovedNOTIFY ( )
    If ABC1.Object.Type = PHASE Then
        ABC1.Object.Top = GTop
    End If
    If ABC1.Object.Type = DEPT Then
        ABC1.Object.Left = GLeft
    End If
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Objects Are Resized

Related

ABC OLE Automation has two events relating to resizing objects: [ObjectSizeSUBCLASS](#) and [ObjectSizedNOTIFY](#).

The [ObjectSizeSUBCLASS event](#) occurs when the user starts to resize an Object object. The [ObjectSizeSUBCLASS event](#) procedure is triggered before ABC initiates any resizing behavior.

The Object to be resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

The [ObjectSizeSUBCLASS](#) example shown below saves the height and width of the object before it is resized in the global variables GHeight and GWidth.

```
Sub ABC1_ObjectSizeSUBCLASS ( )
    GHeight = ABC1.Object.Height      ' Save object height
    GWidth = ABC1.Object.Width        ' Save object width
End Sub
```

The [ObjectSizedNOTIFY event](#) occurs when an Object object is resized. The [ObjectSizedNOTIFY event](#) procedure is triggered after ABC has resized the Object.

The Object that was resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

The [ObjectSizedNOTIFY](#) example below tests the height and width of the resized object and resets these properties to GHeight and GWidth if they are less than those values.

```
Sub ABC1_ObjectSizedNOTIFY ( )
    If ABC1.Object.Height < GHeight Then
        ABC1.Object.Height = GHeight
    End If
    If ABC1.Object.Width < GWidth Then
        ABC1.Object.Width = GWidth
    End If
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Objects Are Deleted

Related

The [DeleteSUBCLASS event](#) occurs when one or more Objects are deleted. The user deletes Objects by selecting the Objects, and then pressing **Del** or choosing Clear from the Edit menu. The [DeleteSUBCLASS event](#) procedure is triggered before ABC performs the deletion.

The Object to be deleted first is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable. You can use the [SelectedObjectCount property](#) of the Chart object to find the number of Objects selected for deletion. You can use the [ItemFromSelection method](#) of the Objects collection in a loop to access the Objects to be deleted.

A [DeleteSUBCLASS](#) example is shown below. This example tests all of the Objects selected for deletion and cancels the selection (and deletion) of any Objects of the DEPT type.

```
Sub ABC1_DeleteSUBCLASS ( )
  Dim Obj As Object                                ' Declare variable

  Do
    Set Obj = Objects.ItemFromSelection             ' Get object to delete
    If Obj.Type = DEPT Then                         ' If DEPT type, then
      Obj.Selected = False                         ' cancel selection
    End If
  Loop While Obj.Valid                             ' Loop until done
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Shapes Are Double Clicked

Related

The [DoubleClickSUBCLASS event](#) occurs when the user double clicks a Shape object. The [DoubleClickSUBCLASS event](#) procedure is triggered before ABC shows the Shape as selected.

The clicked Shape is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

The [DoubleClickSUBCLASS](#) example below beeps to indicate that a shape was double clicked and sets the **Override** property to cancel ABC's standard behavior for the double click event.

```
Sub ABC1_DoubleClickSUBCLASS ( )  
    Beep  
    ABC1.Override = True  
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Shapes Are Drawn Related

The [NewShapeNOTIFY event](#) occurs when the user draws a new Shape object. The [NewShapeNOTIFY event](#) procedure is triggered after ABC draws the Shape.

The newly drawn Shape is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

The [NewShapeNOTIFY](#) example below sets the color of the newly drawn shape to yellow.

```
Sub ABC1_NewShapeNOTIFY ( )  
    ABC1.Object.FillColor = ABC1.App.YELLOW           ' Set color  
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Shapes Are Replaced

Related

The [ReplaceShapeNOTIFY event](#) occurs when the user replaces one or more Shape objects. The [ReplaceShapeNOTIFY event](#) procedure is triggered after ABC replaces the Shape objects.

The Shape to be replaced first is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable. You can use the [ItemFromShapes method](#) of the Objects collection in a loop to access the Shapes to be replaced.

The [ReplaceShapeNOTIFY](#) example below counts the number of Decision shapes in a chart and reports that count to the user.

```
Sub ABC1_ReplaceShapeNOTIFY ( )
    Dim Obj As Object, Objs As Object           ' Declare variables

    ABC.Hourglass = True                       ' Display hourglass
    Counter = 0                                ' Initialize count

    Set Objs = ABC1.Chart.Objects
    Do
        Set Obj = Objs.ItemFromShapes
        If Obj.Shape.ShapeName = "Decision" Then
            Counter = Counter + 1              ' If Decision...
                                                ' ...bump counter
        Loop While Obj                          ' Loop until done

    ABC.Hourglass = False                       ' Clear hourglass
    ABC.MsgBox "Decision Shape Count = " + Counter ' Show results
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Lines Are Drawn

**Related**

The [NewLineNOTIFY event](#) occurs when the user draws a new Line object. The [NewLineNOTIFY event](#) procedure is triggered after ABC draws the Line.

The object to which the newly drawn Line is attached is passed to the event procedure in the Object variable, and the chart in which the Line is located is passed in the Chart variable.

The [NewLineNOTIFY](#) example below sets the color of the object to which the line was just attached to green.

```
Sub ABC1_NewLineNOTIFY ( )  
    ABC1.Object.Color = ABC1.App.GREEN      ' Set color  
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Lines Attach

Related

The [ObjectLineAttachNOTIFY event](#) occurs when the user attaches a line to an Object. The [ObjectLineAttachNOTIFY event](#) procedure is triggered after ABC attaches the Line.

The Object to which the line is attached is passed to the event procedure in the Object variable, the line is passed in the Object2 variable, and the chart in which the Object is located is passed in the Chart variable.

An [ObjectLineAttachNOTIFY](#) example is shown below.

```
Sub ABC1_ObjectLineAttachNOTIFY ( )  
    ABC1.Object.Color = ABC1.App.GREEN      ' Set color  
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Fonts Change

Related

The [ObjectFontChangeNOTIFY event](#) occurs when the user changes the font of one or more Text objects. The [ObjectFontChangeNOTIFY event](#) procedure is triggered after ABC displays the Text objects in the changed font.

The Text object that was changed first is passed to the event procedure in the Object variable, and the chart in which the text is located is passed in the Chart variable. You can use the [ItemFromSelection method](#) of the Objects collection in a loop to access the changed Text objects.

The [ObjectFontChangeNOTIFY](#) example below searches for all TextBlock objects (a Type 2 Text object is a TextBlock object) and resizes them to the same font size when a TextBlock's font size changes.

```
Sub ABC1_ObjectFontChangeNOTIFY ( )
  Dim Text As Object, Objs As Object          ' Declare variables

  ABC.Hourglass = True                       ' Display hourglass

  Set Objs = ABC1.Chart.Objects

  Do
    Set Text = Objs.ItemFromSelection        ' Get selected object
    If Text.Type = 2 Then Exit Do           ' Exit if Type 2
  Loop While Text                            ' Loop until done

  If Text Then                               ' If TextBlock found...
    Objs.ResetSearch                        ' ...reset to first object
    Size = Text.Font.Size                  ' Get font size
    Do
      If Text.Type = 2 Then                 ' If TextBlock object
        Text.Font.Size = Size              ' Set font size
      End If
    Loop While Text                        ' Loop until done

  ABC.Hourglass = False                     ' Clear hourglass
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Field Values Change

Related

The [FieldValueChangedNOTIFY event](#) occurs when the user changes a FieldValue object. The [FieldValueChangedNOTIFY event](#) procedure is triggered after ABC changes the FieldValue.

The FieldValue that was changed is passed to the event procedure in the FieldValue variable, the Object that owns the field is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

A [FieldValueChangedNOTIFY](#) example is shown below. The example tests the new value of the FieldValue object to ensure that it is between 0 and 1000. If the field value is outside of this range, the invalid value is cleared and the user is instructed to enter a value in the valid range.

```
Sub ABC1_FieldValueChangedNOTIFY ( )
    ChangedObject = ABC1.Object
    Message = "Data not in range. Please enter a number between 0 and 1000."
    If ABC1.FieldValue < 0 or > 1000 Then
        ABC1.FieldValue.Empty                ' Clear field value
        ABC1.App.MsgBox Message              ' Display instructions
    End If
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

## When Special Keys Are Pressed

Related

The [SpecialKeySUBCLASS event](#) occurs when the user presses one of the special keys. The [SpecialKeySUBCLASS event](#) procedure is triggered before ABC responds to the key press.

A code representing the key is passed to the event procedure in the **WParam** variable. These codes are defined in the table below.

Key	Code	Key	Code
F1	1	Tab	13
F2	2	Esc	27
F3	3	PgUp	33
F4	4	PgDn	34
F5	5	End	35
F6	6	Home	36
F7	7	Left Arrow	37
F8	8	Up Arrow	38
F9	9	Right Arrow	39
F10	10	Down Arrow	40
F11	11	Ins	45
F12	12	Del	46

A [SpecialKeySUBCLASS](#) example is shown below. It checks for **F11** and **F12**. If **F11** is found, ABC is maximized. If **F12** is found, ABC is minimized.

```
Sub ABC1_SpecialKeySUBCLASS ( )
  If ABC1.WParam = 11 Then
    ABC1.Maximize
    ABC1.Override = True
  End If
  If ABC1.WParam = 12 Then
    ABC1.Minimize
    ABC1.Override = True
  End If
End Sub
```

' If F11...  
' ...maximize ABC and...  
' ...override standard behavior

' If F12...  
' ...minimize ABC and...  
' ...override standard behavior

## Related Topics

### Description

[Handling ABC Events](#)

## When Text Changes

Related

The [ObjectTextChangedNOTIFY event](#) occurs when the user changes a TextBlock object. The [ObjectTextChangedNOTIFY event](#) procedure is triggered after ABC changes the TextBlock.

The Object that owns the text is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

An [ObjectTextChangedNOTIFY event](#) example is shown below. The example displays the text that was changed.

```
Sub ABC1_ObjectTextChangedNOTIFY ( )
    Dim ChangedObject As Object
    Set ChangedObject = ABC1.Object
    ABC1.App.MsgBox "New Text: " + ChangedObject.Text ' Display the new text
End Sub
```

## Related Topics

### Description

[Handling ABC Events](#)

**Related**

## **Working with Chart Files**

**Related**

[Creating New Charts](#)

[Opening Charts](#)

[Saving Charts](#)

[Closing Charts](#)

[Activating a Chart](#)

[Protecting Charts](#)

[Linking Charts](#)

[Linking EXEs to Charts](#)

[Launching Applications](#)

[Printing Charts](#)

[Sending Electronic Mail](#)

[Adjusting the Page Layout](#)

[Displaying Master Items](#)

[Viewing a Chart](#)

[Giving a Presentation](#)

[Using Guidelines](#)

[Defining Measurement Units for a Chart](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Creating New Charts

Related

You can create a new chart using default attributes or attributes that were saved in a template.

In ABC, you create a new chart with default attributes by opening the File menu and choosing New. A new chart window opens. You can create a new chart using the attributes in a specific template by opening the File menu and choosing New From Template. The New From Template dialog box opens, and you can choose the drive, directory, and template file you want to open.

To create a new chart using ABC OLE Automation, you use the [New method](#) or the [NewFromTemplate method](#) of the Application object or the [Add method](#) or the [AddFromTemplate method](#) of the Chart collection.

Use the [New method](#) or the [Add method](#) to create a new chart with default attributes. This opens a new chart window. For example, the following statements each create a new chart, resulting in two new charts.

```
ABC.New  
Charts.Add
```

Use the [NewFromTemplate method](#) or [AddFromTemplate method](#) to create a new chart with attributes based on the chart template's name. For example, the following statements each open a new chart based on the template PORTCOLR.AFT.

```
ABC.NewFromTemplate "c:\abc\samples\PORTCOLR.AFT"  
Chart.AddFromTemplate "c:\abc\samples\PORTCOLR.AFT"
```

There is no practical difference in the effect of [New](#) and [Add](#) or in the effect of [NewFromTemplate](#) and [AddFromTemplate](#).

## Related Topics

### Description

[Working with Chart Files](#)

## Opening Charts

Related

Each chart is stored in a separate file, which contains the shapes, lines, and text in your chart. Chart filenames end with an AF3, AF2, or ABC extension.

In ABC, you open a chart by opening the File menu and choosing Open. The Open dialog box appears, and you can choose the drive and directory that contain the file you want to open.

Using ABC OLE Automation, you open a chart using the [Open method](#) of the Charts collection or the [Open method](#) of the Application object. With each, you specify a fully qualified pathname or partial pathname. If you specify a partial pathname (just the name of the file, for example), the path is the current value of the [DefaultFilePath property](#). If the chart is already open, the [Open method](#) moves the chart to the front. You can optionally specify that the chart is to be opened read only.

For example, the following statement opens the file MYCHART.AF3, located in the path specified in the [DefaultFilePath property](#). The file is opened as read only.

```
ABC.Open "MYCHART.AF3", True
```

[Setting a Default Path for Charts](#)

[Identifying a Chart's Filename](#)

## Related Topics

### Description

[Working with Chart Files](#)

## Setting a Default Path for Charts

Related

You use the [DefaultFilePath property](#) of the Application object to set the default path for all files that are opened or saved. For example, the following statement sets the default path for all files, and then opens a read-only chart without specifying a pathname.

```
ABC.DefaultFilePath = "C:\ABC"  
ABC.Open MYCHART.AF3, True
```

## Related Topics

### Description

[Opening Charts](#)

## Identifying a Chart's Filename

Related

You can find a chart's filename with or without its pathname. The [FullName property](#) of the Chart object returns the fully qualified pathname of the chart. (If the chart has not been saved, it returns the temporary name of the chart.) The [Name property](#) of the Chart object returns the name of the chart without the path. You access the information about charts using the [Charts property](#) of the Application object.

For example, the following statements return the fully qualified pathname and the name without the path of a chart.

```
FullyQualifiedPathname = Chart.FullName  
NameOnly = Chart.Name
```

## Related Topics

### Description

[Opening Charts](#)

## Saving Charts

Related

In ABC, when you save a chart, ABC stores the chart in a file on disk. Each chart is saved in a separate file. When you save a chart, you can assign it a name and choose where you want to store it on a disk.

In ABC OLE Automation, you can find if the chart has been saved to disk and if the file on disk is the same as the file in memory. You can save the current version of the chart to a specified or default pathname.

You use the [HasDiskFile property](#) of the Chart object to find out if the chart has ever been saved to disk. For example, the following statement puts into the variable EverSaved whether the current chart has a file on disk.

```
EverSaved = Chart.HasDiskFile
```

You can use the [Saved property](#) of the Chart object to find if the file saved on disk is the same chart as currently resides in memory. If the value of the [Saved property](#) is True, there is no need to save the chart. For example, the following statement puts into the variable FileCurrent whether the current chart has been changed since it was last changed.

```
FileCurrent = Chart.Saved
```

You use the [Save method](#) of the Chart object to save a file. You can optionally specify a path and filename. If you only specify a filename, the pathname is the value of the [DefaultFilePath property](#).

You also can optionally specify the type of file to save it as. The following table shows the types of files possible.

File Type	Save File As
0	Chart, version 3.0
1	Template, version 3.0
2	Chart, version 2.0
3	Template, version 2.0

For example, the following saves a file as a template with the name MYTEMPL.AF3.

```
Chart.Save "MYTEMPL.AF3", 1
```

You can use the [HasDiskFile property](#), [Saved property](#), and [Save method](#) together to save a file only when necessary. For example, the following statements save a file with a new name if it has never been saved, or save it with its current name if it has been changed since it was last saved.

```
EverSaved = Chart.HasDiskFile           ' Is chart on disk?
If Not EverSaved Then                   ' If not, save it
    NextFile = "File" + ChartFileCount + ".AF3"
    ABC.Hint = "Saving chart as " + NextFile + "."
    Chart.Save NextFile, 0               ' Create filename
    ChartFileCount = ChartFileCount + 1
Else                                     ' Increment file counter
    FileCurrent = Chart.Saved            ' Else, check if changed
    If Not FileCurrent Then               ' If changed, save it
        ABC.Hint = "Saving chart."
        Chart.Save
    End If
End If
```

[Reverting to the Last Saved Version](#)  
[Read-Only Charts](#)

## Related Topics

### Description

[Working with Chart Files](#)

## Reverting to the Last Saved Version

Related

You use the [RevertToSaved method](#) of the Chart object to revert to the last saved copy of the document, discarding any changes. For example, the following statement removes the current version of the chart from memory and opens the version on the hard disk.

```
Chart.RevertToSaved
```

## Related Topics

### Description

[Saving Charts](#)

## Read-Only Charts

Related

Some charts are opened as read-only charts, either because they were opened that way using the [Open method](#) or because the person does not have rights to save the chart under the same filename. You can determine whether a chart is read only using the [ReadOnly property](#) of the Chart object. For example, the following statement puts into the variable `ReadOnlyFile` whether the user can save the file under its current path and filename.

```
ReadOnlyFile = Chart.ReadOnly
```

The [ReadOnly property](#) is read only. You cannot change its value. To open a file as read only, use the optional *AsReadOnly* parameter of the [Open method](#) of the Charts collection or of the Application object.

## Related Topics

### Description

[Saving Charts](#)

## Closing Charts

Related

You can close just the active chart or close all of your charts at once. In ABC, you close the open charts in the order that they are arranged on the screen. If the open chart contains changes that you have not saved, ABC displays a message asking if you want to save the changes.

In ABC OLE Automation, you use the [CloseChart method](#) of the Chart object to close the chart. **Note:** When you use the [CloseChart method](#), the user does not get a prompt to save the chart. For example, the following statement closes a chart without any prompt to the user.

```
Chart.CloseChart
```

### Closing All Charts at Once

In ABC, you close all the open charts by opening the File menu and choosing Close All. If any of the charts contain unsaved changes, ABC asks if you want to save the changes.

In ABC OLE Automation, you use the [CloseAll method](#) of the Application object or the [CloseAll method](#) of the Charts collection to close all charts in the ABC workspace. **Note:** If any of the charts contain unsaved changes, ABC does not ask if you want to save the changes.

For example, the following two statements each close all the open charts.

```
ABC.CloseAll  
ChartCollection.CloseAll
```

## Related Topics

### Description

[Working with Chart Files](#)

## Activating a Chart

Related

Activating a chart lets you return a previously created chart. In ABC, you activate a chart by clicking on it or by opening the Window menu and choosing the chart from the numbered list of open charts.

In ABC OLE Automation, you can activate a chart using the [Activate method](#) of the Application object and the [Item method](#) of the Charts collection. You can find the current active chart using the [ActiveChart property](#) of the Application object.

The [Item method](#) lets you identify the chart you want to bring to the front. It takes one parameter, which is either a string indicating the full path and executable name of the chart or a number that is the chart's ordering position within the collection. For example, the following statement brings the chart C:\ABC\MYCHART.AF3 to the front and places the chart object in ActiveChart.

```
ActiveChart = ABC.Charts.Item("C:\ABC\MYCHART.AF3").Activate
```

If the chart is not open, the method returns a nonvalid chart object.

The [Count property](#) of the Charts collection contains the number of charts in the collection. (The [Count property](#) exists in several collections. All of them work approximately the same way.) You can use the [Count property](#) to loop through the open charts. For example, the following statements search through the chart collection looking for the chart MYCHART.AF3 and bring it to the front when it is found.

```
For ChartCount = 1 to ChartsCollection.Count           ' Search collection
  Set CurrentChart = ABC.Item(ChartCount)
  If CurrentChart.Name = "MYCHART.AF3" Then
    Exit For                                           ' Exit when chart is found
  End If
Next ChartCount
CurrentChart.Activate                                 ' Activate chart
```

## Related Topics

### Description

[Working with Chart Files](#)

## Protecting Charts

Related

At times, you may want to prevent other people from editing your chart. After you assign a password to protect the chart, no one is able to move, edit, add, or delete objects in the chart until they enter the password correctly.

In ABC, you can use password protection to manage linked files. By assigning each person in a work group a different password, you can ensure that each person has access to make changes only to his or her own charts.

In ABC OLE Automation, you use the [SetProtection method](#) of the Chart object to turn the protection on and off. The [SetProtection method](#) has two parameters. The first is a Boolean that turns protection on and off. The second is the password. For example, the following statements turn protection on, and then turn it back off.

```
Chart.SetProtection True, "Quint"  
Chart.SetProtection False, "Quint"
```

You use the [Protected property](#) of the Chart object to identify whether a chart is protected. The [Protected property](#) is read only, so you cannot use it to change the protection of a chart.

For example, the following statements turn off password protection for a chart if the chart is protected.

```
If CurrentChart.Protected Then  
    CurrentChart.SetProtection False, CurrentChartPassword  
End If
```

## Related Topics

### Description

[Working with Chart Files](#)

## Linking Charts

Related

You can link charts together. After the charts are linked, you can double click a designated shape in one chart to open the linked chart automatically.

[Linking Shapes to Other Charts](#)

[Creating Group Links](#)

[Opening a Linked Chart](#)

[Choosing Link Indicators](#)

## Related Topics

### Description

[Working with Chart Files](#)

## Linking Shapes to Other Charts

Related

In ABC, you link an object, such as a shape, to an active chart by clicking the Object Selector tool in the toolbox, selecting the shape you want to link to another chart, then clicking the Link button in the ribbon. In the Link dialog box you identify the chart to which you want to link and choose an action.

In ABC OLE Automation, you can link shapes to other charts, determine if a shape is linked to another chart, and link to field data from another chart.

You use the [LinkedChartName property](#) of the Shape object to provide the full pathname of a chart linked to an object and link the shape to the file. For example, the following statement links a shape to a chart.

```
MyShape.Shape.LinkedChartName = "C:\ABC\LINKCHT.AF3"
```

You use the [IsLinked property](#) of the Shape object to find if a shape is linked to another chart. The [IsLinked property](#) is read only. The property returns True if the object contains a link to another chart.

```
ShapeLinked = MyShape.Shape.IsLinked
```

You use the [LinkFields property](#) of the Shape object to accumulate the linked chart's field data into the shape's field information, if the shape is linked to another chart with field information. For example, the following statement turns on putting field data from the linked chart into the shape's field data.

```
MyShape.Shape.LinkFields = True
```

## Related Topics

### Description

[Linking Charts](#)

## Creating Group Links

Related

The group and link function lets you move a group of selected objects to another chart, and replace the moved group with a shape that is linked to the chart to which the group was moved.

In ABC, you group and link by clicking the Object Selector tool in the toolbox, selecting the objects you want to move to another chart, then clicking the Link button in the ribbon. In the Link dialog box you identify the chart to which you want to link, and choose Group and Link. The moved group is replaced by the currently selected shape in the Shape Palette.

In ABC OLE Automation, you group and link with the [GroupAndLink method](#) of the Chart object.

The [GroupAndLink method](#) returns the shape that replaced the moved group and has two optional parameters. The first parameter specifies the full pathname of the new chart. If the first parameter is omitted, ABC generates a default chart pathname. The second parameter specifies whether the new chart's fields are linked to the source chart. If the second parameter is omitted, ABC does not link the fields. Use a variable with a True value for the second parameter to link the fields.

After executing [GroupAndLink](#), you can obtain the newly created chart object with the [ActiveChart property](#) of the Application object.

The example shown below moves the selected objects to a chart named LINKCHT.AF3. The LINKCHT.AF3 fields are not linked to the source chart.

```
Set ShapeGroupLink = GroupAndLink("C:\ABC\LINKCHT.AF3")
```

## Related Topics

### Description

[Linking Charts](#)

## Opening a Linked Chart Related

You can use a linked shape to open a linked chart. In ABC, you open a linked chart by double clicking the linked shape. The linked chart opens and becomes the active chart.

In ABC OLE Automation, you use the [Link method](#) of the Shape object to open the linked chart. For example, the following statement opens the chart attached to a shape and puts the linked chart into LinkedChart.

```
Set LinkedChart = MyShape.Shape.Link
```

**Note:** If there is no value in the [LinkedChartName property](#), using the [Link method](#) creates a new chart with an automatically generated filename.

## Related Topics

### Description

[Linking Charts](#)

## Choosing Link Indicators Related

The link indicators appear in linked shapes. You can specify indicators for the linked shapes and place a shadow on objects with linked files.

In ABC OLE Automation, use the [LinkIndicator property](#) of the Chart object to specify the indicator, up to three characters, used for linked shapes. Use the [LinkShadow property](#) of the Chart object to show a shadow on shape objects that have linked files. For example, the following statements specify a link indicator of LNK and show a shadow.

```
Chart.LinkIndicator = "LNK"  
Chart.LinkShadow = True
```

## Related Topics

### Description

[Linking Charts](#)

## Linking EXEs to Charts

Related

You can link a compiled Visual Basic EXE program file to a chart so that the EXE program runs automatically when you open the chart. This feature is illustrated by the Deployment Wizard sample program shipped with ABC OLE Automation. The Deployment Wizard automatically links DEPLOY.EXE to every new chart that you create with the Deployment Wizard.

You use the `TypeRequiresEXE` and `TypeUsesEXE` properties of the Chart object to link an EXE to a chart.

- If you set the [TypeRequiresEXE property](#) to True, the chart requires the EXE to open. If the linked EXE cannot be run, then the chart does not open.
- If you set the [TypeUsesEXE property](#) to True, then the chart attempts to run the linked EXE when it opens. If the EXE cannot be run, the chart still opens, after ABC displays a warning.

The name of the EXE that is linked to a chart by these properties is determined by the [Type property](#) of the chart. The EXE name is constructed by adding .EXE to the chart [Type](#). In the case of the Deployment Wizard sample program, for example, the chart [Type](#) is DEPLOY. Therefore, the EXE linked to a new chart created by the Deployment Wizard is DEPLOY.EXE.

The following sample code specifies the chart [Type](#) and links an EXE to the chart. If the `CHARTTYPE` variable is set to "DEPLOY," then this code sample links the chart to DEPLOY.EXE.

```
Chart.Type = CHARTTYPE  
Chart.TypeRequiresEXE = True
```

If you set either [TypeRequiresEXE](#) or [TypeUsesEXE](#) to True in a program, then you also must ensure that you close all charts of that [Type](#) when your program closes. You use the [ChartTypeShutdown method](#) of the Application object to close the charts. The following code sample, located in the `Form.QueryUnload` procedure of your program, closes all charts of the [Type](#) `CHARTTYPE`.

```
ABC.ChartTypeShutdown CHARTTYPE, APPNAME
```

**Note:** ABC only runs one instance of a linked EXE. When a second chart that is linked to an already running EXE is loaded, ABC refers to the currently running EXE. It does not load a second copy of the EXE.

## Related Topics

### Description

[Working with Chart Files](#)

## Launching Applications

Related

You can launch other Windows applications from within ABC. Launching lets you open other applications without using the Program Manager. It also lets you easily send information about a chart and shape to an application, such as a database, that is prepared to receive it.

[Setting Shapes to Launch Applications](#)

[Launching Applications](#)

[Choosing Launch Indicators](#)

## Related Topics

### Description

[Working with Chart Files](#)

## Setting Shapes to Launch Applications

Related

To set a shape to have a launch, you enter the program it is to launch. To launch an application, you use shapes in the Chart object with attached launches.

In ABC, you set a shape to launch an application by clicking the Object Selector tool in the toolbox, selecting the shape you want to use, and clicking the Launch button in the ribbon. In the Launch dialog box, you specify the command line, directory, and flags.

In ABC OLE Automation, you use the [LaunchCommand property](#) of the Shape object to set a command to launch for the object. You use the [LaunchStartDir property](#) of the Shape object to specify the starting directory for the launch. You use the [LaunchFlags property](#) of the Shape object to specify the flags for executing a launch.

The [LaunchFlags property](#) can take a combination of flags. The parameter for the property is the sum of the values. The possible settings values for the flags are listed in the table below.

Flag Value	Description
1	Chart name
2	Shape number
4	Shape text

For example, the following statements set a shape to launch Excel, set the starting directory to a subdirectory of Excel, and set the flags to pass the chart name and shape text.

```
CurrentChart.Shape.LaunchCommand = "C:\EXCEL\EXCEL.EXE"  
CurrentChart.Shape.LaunchChartDir = "C:\EXCEL\MYCHARTS"  
CurrentChart.Shape.LaunchFlags = 5
```

## Related Topics

### Description

[Launching Applications](#)

## Launching Applications

Related

You use the shape you set for launching to launch the application. The launch indicators appear in shapes set for launching.

In ABC, to launch an application, you press and hold **Ctrl** and double click the shape you set for launching.

In ABC OLE Automation, you use the [Launch method](#) of the Shape object to execute the shape's launch. You identify whether a shape is set to launch an application using the [IsLaunched property](#) of the Shape object. For example, the following statements check to see if a shape has a launch and, if it does, they launch the application and put a Boolean value in LaunchSuccessful to indicate whether the launch succeeded.

```
If CurrentChart.Shape.IsLaunched Then
    LaunchSuccessful = CurrentChart.Shape.Launch
End If
```

## Related Topics

### Description

[Launching Applications](#)

## Choosing Launch Indicators

Related

The launch indicators appear on shapes with attached launches. You can specify indicators for the launch shapes and place a shadow on those objects.

In ABC OLE Automation, use the [LaunchIndicator property](#) of the Chart object to indicate which shapes have launches attached. Use the [LaunchShadow property](#) to show a shadow on shape objects that are associated with a launch. For example, the following statements specify a launch indicator of LCH and show a shadow.

```
Chart.LaunchIndicator = "LCH"  
Chart.LaunchShadow = True
```

## Related Topics

### Description

[Launching Applications](#)

## Printing Charts

Related

In ABC, when you open the File menu and choose the Print command, the Print dialog box opens. You can print all the pages, a range of pages, or the selected objects in the chart.

In ABC OLE Automation, you use the [PrintOut method](#) of the Chart object to print the chart object. The parameters for the [PrintOut method](#) specify the options to use when printing.

Parameter	Description
FromPage	Integer (default is page 1)
ToPage	Integer (default is last page)
NumberOfCopies	Integer (default is 1)
FitToPage	Integer (Boolean) (default is False)
PrintNotes	Integer (Boolean) (default is False)

For example, the following statement prints a chart using the default parameters.

```
Chart.PrintOut
```

You use the [PrintSelected method](#) of the Chart object to print the selected objects in the chart. The parameters for the [PrintSelected method](#) indicate the options to use when printing.

Parameter	Description
NumberOfCopies	Integer (default is 1)
FitToPage	Integer (Boolean) (default is False)
PrintNotes	Integer (Boolean) (default is False)

For example, the following statement prints the selected objects in a chart using the default parameters.

```
Chart.PrintSelected
```

You use the [Printer property](#) of the Application Object to specify the current printer, the one to use when printing. When you read the value of the [Printer property](#), it returns the current printer and port. For example, it might return "HP LaserJet III on LPT2:."

When you set the value, the program uses a "loose matching" routine so, for example, setting the [Printer property](#) to "HP Laser" or "LPT2" chooses "HP LaserJet III on LPT2:" if that is the printer on LPT2:. If more than one printer matches the value you set, the exact match is used first. If there is not exact match, the first one alphabetically is used. For example, the following statement sets the printer to the first printer available on LPT1:.

```
ABC.Printer = "LPT1:"
```

You use the [PrintBlankPages property](#) of the PageLayout object to specify whether a blank page should be printed if there are no objects on the page. For example, the following statement will print pages even if they are blank when you use the [PrintOut method](#).

```
Chart.Layout.PrintBlankPages = True
```

## Related Topics

### Description

[Working with Chart Files](#)

## Adjusting the Page Layout Related

Page layout options affect the orientation and dimensions of the pages in a chart. To adjust the page layout, you specify the object, then specify the properties of the drawing area and page.

In ABC, you open the File menu and choose the Page Layout command. The Page Layout dialog box opens. You use this dialog box to choose paper size, orientation, page margins, units of measure, print order, and whether to print blank pages.

In ABC OLE Automation, you can specify the object and various options. The following table shows the properties of the PageLayout object and their meanings. You use the [PageLayout property](#) of the Chart object to access the PageLayout object.

<b>Property</b>	<b>Meaning</b>
<a href="#">Height</a>	Height of the drawing area
<a href="#">Width</a>	Width of the drawing area
<a href="#">MarginBottom</a>	Bottom margin of the page
<a href="#">MarginLeft</a>	Left margin of the page
<a href="#">MarginRight</a>	Right margin of the page
<a href="#">MarginTop</a>	Top margin of the page
<a href="#">Orientation</a>	Portrait (0) or landscape (1)
<a href="#">PageHeight</a>	Height of the page
<a href="#">PageWidth</a>	Width of the page
<a href="#">PageOrder</a>	Across, then down  (0) or down, then across  (1)
<a href="#">PaperSize</a>	Size of the paper to be printed

The [PaperSize property](#) uses a "loose matching" routine when you are setting the value. For example, setting the [PaperSize property](#) to "let" chooses "Letter 8 1/2 x 11 in."

You use the [PageCount property](#) of the Chart object to set the number of pages in a chart.

For example, the following statements set the drawing height to 8.5 inches, the drawing width to 11 inches, the margins to 0.5 inches on all sides, the orientation to landscape, the page height to 8.5 inches, the page width to 11 inches, the page order to across-then-down, and the paper size to "Letter 8 1/2 x 11 in" and sets the chart to have four pages.

```
Chart.Layout.Height = 8.5
Chart.Layout.Width = 11
Chart.Layout.MarginBottom = .5
Chart.Layout.MarginLeft = .5
Chart.Layout.MarginRight = .5
Chart.Layout.MarginTop = .5
Chart.Layout.Orientation = 1
Chart.Layout.PageHeight = 8.5
Chart.Layout.PageWidth = 11
Chart.Layout.PageOrder = 0
Chart.Layout.PaperSize = "Letter"
Chart.PageCount = 4
```

## Related Topics

### Description

[Working with Chart Files](#)

## Displaying Master Items

Related

You can define and display useful pieces of information in a chart by displaying Master Items.

In ABC, you open the File menu and choose the Master Items command. The Master Items dialog box opens. In this dialog box you choose whether to display the chart name, page numbers, the date and time, and a logo. You also can enter one or two text lines, possibly for use as header and footer text, can choose the format for the date, and can choose whether master items appear on the first page or on all pages.

In ABC OLE Automation, you can specify the same information using properties of the MasterItems object. You access the information about master items using the [MasterItems property](#) of the Chart object. The following table shows the properties of the MasterItems object and their meanings.

<b>Property</b>	<b>Meaning</b>
<a href="#">ChartName</a>	Chart name master item object
<a href="#">ChartNameShown</a>	Whether the chart name master item is shown (Boolean)
<a href="#">Date</a>	Date master item object
<a href="#">DateShown</a>	Whether the date master item is shown (Boolean)
<a href="#">DateStyle</a>	MM/DD/YY (0), short text (1), long text (2)
<a href="#">Logo</a>	Logo master item (the Logo property is read only, but the properties from the object it returns are read/write)
<a href="#">LogoPathname</a>	Fully qualified pathname of the logo
<a href="#">LogoShown</a>	Whether the logo master item is shown (Boolean)
<a href="#">PageNumber</a>	Page number master item object
<a href="#">PageNumberShown</a>	Whether the page number master item is shown (Boolean)
<a href="#">Range</a>	First page only (0) or all pages (1)
<a href="#">Text1</a>	Text1 master item (the Text1 property is read only, but the properties from the object it returns are read/write)
<a href="#">Text1Shown</a>	Whether the text1 number master item is shown (Boolean)
<a href="#">Text2</a>	Text1 master item (the Text2 property is read only, but the properties from the object it returns are read/write)
<a href="#">Text2Shown</a>	Whether the text2 number master item is shown (Boolean)
<a href="#">Time</a>	Time master item object
<a href="#">TimeShown</a>	Whether the time master item is shown (Boolean)

The following table shows the methods of the MasterItems object and their meanings.

<b>Method</b>	<b>Meaning</b>
<a href="#">HideAll</a>	Hide all master items
<a href="#">ShowAll</a>	Show all master items
<a href="#">UpdateDateAndTime</a>	Update the date and time to the system date and time or to a specified date and time

The following statements show the date and time, with the date in long text format, and show a first line of text in bold.

```
Chart.MasterItem.Date.DateShown = True
Chart.MasterItem.Date.DateStyle = 2
Chart.MasterItem.Time.TimeShown = True
Chart.MasterItem.Text1.Text = "First line of text."
Chart.MasterItem.Text1.Bold = True
Chart.MasterItem.Text1.Text2Shown = True
```

## Related Topics

### Description

[Working with Chart Files](#)

## Viewing a Chart Related

Scrolling through a chart lets you display all areas of the chart. In ABC OLE Automation, you can scroll through a chart by specifying the left and top points in the chart. You can scroll to a specific page or location.

You use the [ScrollLeft property](#) of the Chart object to set the left point visible in the chart and the [ScrollTop property](#) of the Chart object to set the top point visible in the chart.

You use the [ScrollPage method](#) of the Chart object to scroll the chart to a particular page and the [ScrollPosition method](#) of the Chart object to scroll to a location in the chart by specifying a vertical and horizontal position.

You use the [View property](#) of the Chart object to view a particular page of the document. The following table shows the parameters for the [View property](#) and their meanings.

Value	Description
0	One to one
1	Current page
2	Used pages
3	Percentage zoom

For example, the following statements change the view to show the current page and then go to the second page in the chart.

```
Chart.View = 1  
Chart.ScrollPage 2
```

You use the [ZoomPercentage property](#) of the Chart object to change the magnification of the current document. You can set the view to any value from 25% to 400% of the actual size of the objects in the chart.

For example, the following statement changes the view to show the chart at 200% of its actual size

```
Chart.ZoomPercentage = 200
```

## Related Topics

### Description

[Working with Chart Files](#)

## Giving a Presentation

Related

The Full Screen feature in ABC lets you show charts as "slides" in a presentation easily, without the distracting menus and buttons. The Full Screen command is in the Windows menu or the View menu. With ABC OLE Automation, you use the [FullScreen method](#) of the chart object. Use the [CancelFullScreen method](#) of the chart object to return the chart to the previous view.

In ABC, you can use linked charts to move from one "slide" (chart) to another by double clicking the linked shape. With ABC OLE Automation, you can give a slide show by showing successive charts at the full screen view, delaying each one for a few seconds.

For example, the following statements show two charts on the full screen, with an appropriate delay routine between them, and then return to the previous view.

```
Chart1.FullScreen  
[Delay routine]  
Chart2.FullScreen  
Chart2.CancelFullScreen
```

## Related Topics

### Description

[Working with Chart Files](#)

## Using Guidelines

Related

You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline. Guidelines let you align shapes of different sizes for an attractive, organized look.

In ABC, you drag guidelines from the rulers. If the Align to Rulers option is selected in the Preferences dialog box, guidelines snap to ruler position. The guidelines do not appear in the printed chart.

In ABC OLE Automation, you can toggle the guidelines, add horizontal and vertical guidelines, and clear all guidelines.

You use the [GuidelinesOn property](#) of the Chart object to turn showing guidelines on and off. You use the [AddHorizontalGuideline method](#) of the Chart object to add a horizontal guideline at the vertical position passed. Use the [AddVerticalGuideline method](#) of the Chart object to add a vertical guideline at the horizontal position passed. Use the [ClearGuidelines method](#) of the Chart object to remove all guidelines from the chart.

For example, the following statements create a horizontal guideline four inches down from the top, create a vertical guideline three inches over from the left, turn showing guidelines on, and then remove all guidelines from the chart.

```
Chart.AddHorizontalGuideline 4  
Chart.AddVerticalGuideline 3  
Chart.GuidelinesOn = True  
Chart.ClearGuidelines
```

## Related Topics

### Description

[Working with Chart Files](#)

## Defining Measurement Units for a Chart

Related

The measurement units for a chart specify the size and distance values. In ABC, you specify the measurement units for a chart by opening the File menu and choosing Preferences.

In ABC OLE Automation, you use the [Units property](#) of the Chart object to specify the units for measurement in the chart object and all its child chart objects. In addition, the [Units property](#) specifies the size and distance values passed in the Preferences object. The default unit value is 0 (inches) for each new Preferences object.

Value	Description
0	Inches
1	Centimeters

For example, the following statement sets the measurement unit to centimeters.

```
Chart.Units = 1
```

## Related Topics

### Description

[Working with Chart Files](#)

## Sending Electronic Mail

Related

In ABC you can attach the current chart to an e-mail message so you can send it using a MAPI e-mail system such as Microsoft Mail. You bring up the e-mail system by choosing Send Mail in the ABC File menu.

In ABC OLE Automation, you use the [SendMail method](#) of the Chart object to create a new e-mail message with a Chart object attached. The user must then address the mail to the appropriate person and can add a message.

For example, the following statement creates a new e-mail message with the chart object as an attachment.

```
Chart.SendMail
```

## Related Topics

### Description

[Working with Chart Files](#)

Related

## Working with Objects

Related

[Identifying an Object](#)

[Finding the Number of Items](#)

[Finding Objects in a Chart](#)

[Selecting Objects in a Chart](#)

[Moving Objects](#)

[Resizing Objects](#)

[Changing the Display Order of Objects](#)

[Setting the Current Drawing Position](#)

[Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#)

[Using OLE Client Objects](#)

[Speeding Actions](#)

[Undoing Actions](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Identifying an Object

Related

You can identify objects using the [Type](#) and [UniqueID](#) properties of the Object object and the [ShapeName property](#) of the Shape object. You access the information about objects using the [Objects property](#) of the Chart object.

The Type property contains a value that specifies the type of the object. You cannot change the value of the Type property. The following table shows the values and their meanings.

Value	Meaning
0	Shape
1	Line
2	Text
3	Bitmap
4	OLE client object
5	Master item

By determining the type of object, you can limit the operations you perform on it. For example, if the value of the [Type property](#) is 1 (line), then you would not set the font size.

The [UniqueID property](#) returns a unique identifier that you can use to choose an object in the Objects collection using, for example, the [ItemFromUniqueID method](#). The identifier is unique for each object in each chart. If you wish, you could create a database containing the [UniqueID property](#) values for all the objects in a chart to make it easy to identify and act on each of them.

The [ShapeName property](#) contains the name of the shape, such as "Process" or "Decision." You cannot change the value of the [ShapeName property](#).

After you find the type of shape for the shape you want, you could, for example, use the information to create another shape of the same type. The following statements find the name of the current shape and then create another shape of the same type using the [DrawShape method](#).

```
CurrentShape = Shape.ShapeName  
Set SameShape = Chart.DrawShape(CurrentShape)
```

## Related Topics

### Description

[Working with Objects](#)

## Finding the Number of Items

Related

You can use ABC OLE Automation to find how many objects there are and how many objects of different types are selected.

[Finding the Total Number of Objects](#)

[Finding the Number of Objects Selected](#)

## Related Topics

### Description

[Working with Objects](#)

## Finding the Total Number of Objects

Related

You can find how many objects there are using the [Count property](#) of the Objects collection. For example, you can use this property to post a message in the hint line telling how many objects are in a chart. You cannot change the value of the [Count property](#).

The following statement sets ObjectCount to the number of objects in a chart.

```
ObjectCount = ABC.ActiveChart.Objects.Count
```

When you know how many objects are in a chart, you can specify them by number using the [Item method](#) in the Objects collection. For example, the following statements turn all the objects in a chart green.

```
Dim ChartObjects As object
Set ABC = CreateObject("ABCFlow.application")
Set ChartObjects = ABC.ActiveChart.Objects
For ItemCount = 1 to ChartObjects.Count
    ChartObjects.Item(ItemCount).Color = ABC.GREEN
Next ItemCount
```

**Note:** The statements above turn the objects, including TextBlocks, but not the text inside a shape, green. To turn the text inside a shape green, use the [Color property](#) described in [Setting Text Colors](#).

## Related Topics

### Description

[Finding the Number of Items](#)

## Finding the Number of Objects Selected

Related

Sometimes it is useful to know the number of objects that are selected in a chart. You can use the [SelectedObjectCount](#), [SelectedShapeCount](#), [SelectedLineCount](#), and [SelectedOtherCount](#) properties in the Chart object to find out how many objects in a chart are selected. You cannot change the values of any of these properties except by selecting or deselecting objects.

The [SelectedObjectCount property](#) contains the total number of selected objects in the chart. It equals the sum of the values of the [SelectedShapeCount](#), [SelectedLineCount](#), and [SelectedOtherCount](#) properties.

For example, the following statement sets TotalSelected to the number of objects in the chart.

```
TotalSelected = Chart.SelectedObjectCount
```

The [SelectedLineCount property](#) contains the number of selected lines, not the number of selected line segments, so the routing of the lines does not affect the count.

The [SelectedShapeCount property](#) contains the number of selected shapes.

The [SelectedOtherCount property](#) contains the number of objects selected that are not shapes or lines. It includes TextBlock objects, master item objects such as the date and headers, OLE objects, bitmaps, and other objects pasted into ABC.

## Related Topics

### Description

[Finding the Number of Items](#)

## Finding Objects in a Chart

Related

You can use the ItemFrom methods to find one or more objects in a chart that meet a criterion.

You use the [ItemFromAll](#), [ItemFromShapes](#), [ItemFromLines](#), [ItemFromSelection](#), [ItemFromText](#), [ItemFromFieldValue](#), [ItemFromAttachments](#), [ItemFromNumber](#), and [ItemFromUniqueID](#) methods in the same way. These methods are all in the Object Collections object. The following table shows the parameters you specify with each of the methods, the object that it returns, and an example of the method.

Method	Parameters	Return Object	Example
<a href="#">ItemFromAll</a>	None	Object	ItemFromAll()
<a href="#">ItemFromShapes</a>	None	Shape object	ItemFromShapes()
<a href="#">ItemFromLines</a>	None	Line object	ItemFromLines()
<a href="#">ItemFromSelection</a>	None	Selected object	ItemFromSelection()
<a href="#">ItemFromText</a>	Text	Object containing text	ItemFromText("Buy")
<a href="#">ItemFromFieldValue</a>	Field template object and value	Object with value in the field	ItemFromFieldValue (Field1,1200)
<a href="#">ItemFromAttachments</a>	One or two objects	Attached shape, text, or line object	ItemFromAttachments (NewObj1,NewObj2)
<a href="#">ItemFromNumber</a>	Shape number	Shape with the number	ItemFromNumber(3)
<a href="#">ItemFromUniqueID</a>	Unique ID	Object with that ID	ItemFromUniqueID (7)

The [ItemFromAll](#), [ItemFromShapes](#), [ItemFromLines](#), and [ItemFromSelection](#) methods do not take any parameters. They return all objects, shape objects, line objects, and selected objects, respectively.

The [ItemFromText method](#) returns objects that contain the text you specify.

The [ItemFromFieldValue method](#) returns objects with the value in the specified field.

The [ItemFromAttachments method](#) returns the objects that are attached to the one or two objects you specify. For example, if you specify two shapes, this method would return the line connecting them.

The [ItemFromNumber method](#) returns the shape with the number you specify.

The [ItemFromUniqueID method](#) returns the object with the unique identifier you specify. You can find the identifier using the [UniqueID property](#) of the Object object.

For example, the following statement sets the unique identifier of an object into the variable CurrentID.

```
CurrentID = NewObj1.UniqueID
```

The [Valid property](#), found in the Chart object, contains a Boolean value based on whether the current object is valid or not. You normally use the [Valid property](#) in the While portion of a Do While loop to ensure that only valid objects are used. **Note:** By default, the current object is valid unless set otherwise.

To use the ItemFrom methods, you use them in a loop, most often a Do While loop. Each time the loop executes, the method returns the next object, so you can test the objects for a property value and act on the objects that meet that value. For example, the following changes to red all shapes that have the word "Buy" in them.

```
Sub Form_Load ()
    Dim ABC As object
    Dim ChartObjects As object
    Dim TestObject As object
    Set ABC = CreateObject("ABCFlow.application")
    Set ChartObjects = ABC.ActiveChart.Objects
```

```
Do
    Set TestObject = ChartObjects.ItemFromText("Buy")
    TestObject.Shape.FillColor = ABC.RED
Loop While TestObject.Valid
End
End Sub
```

**Note:** If you change the line `TestObject.Shape.FillColor = ABC.RED` to `TestObject.Color = ABC.RED`, this example also turns master item text to red if it contains the word "Buy."

If you wish, you can reset all searches to start at the beginning of the items in the chart using the [ResetSearch method](#), found in the Objects collection.

```
ChartObjects.ResetSearch
```

## Related Topics

### Description

[Working with Objects](#)

## Selecting Objects in a Chart Related

The ItemFrom methods described in the previous section of this chapter let you identify objects in a chart so you can make changes to them, but they do not select them, and only the [ItemFromSelection method](#) makes any note of whether objects are selected.

You can use the [Selected property](#) in the Object object to determine if an object is selected. For example, the following statements turn an object black if it is selected.

```
If NewObj1.Selected Then
    NewObj1.Color = ABC.BLACK
End If
```

You also can use the [Selected property](#) to select an object. For example, the following statement selects the specified object.

```
NewObj1.Selected = True
```

You can use the [Select method](#) of the Chart object to select and deselect a group of objects. The following table shows the action of the values.

Value	Action
0	Selects all shapes in addition to anything already selected
1	Selects all lines in addition to anything already selected
2	Selects everything
3	Deselects everything

Values of 0, 1, and 2 in the [Select method](#) are the equivalent of opening the Select submenu in the ABC Edit menu and choosing Shapes, Lines, or All.

You can use the [SelectShapeType method](#) in the Chart object to select all shapes of a specific type. For example, the following statement selects all Decision (diamond) shapes in addition to any objects already selected.

```
Chart.SelectShapeType("Decision")
```

You can deselect all objects using the [DeselectAll method](#) of the Chart object. The [DeselectAll method](#) has the same effect as the [Select method](#) with a value of 3.

```
Chart.DeselectAll
```

## Related Topics

### Description

[Working with Objects](#)

## Moving Objects

Related

You move objects using the [Top](#), [Bottom](#), [Left](#), [Right](#), [CenterX](#), and [CenterY](#) properties of the Object object. You also can use those properties to find the location of an object.

**Note:** These properties move objects, but do not resize them. Resize objects with [Width](#) and [Height](#).

The [Top](#), [Bottom](#), [Left](#), and [Right](#) properties describe the location of the specified side of the object. The [CenterX](#) and [CenterY](#) properties describe the horizontal and vertical positions of the center of the object. For example, the following statements set the top of the object to two inches from the top of the page and set the center of the object three inches from the left side of the page.

```
Object.Top = 2  
Object.CenterX = 3
```

The following statements check to see if the center of the object is within one inch of the upper left corner of the page. If it is, the object is moved so its bottom and left sides are two inches from the top and left side of the page.

```
If Object.CenterX <= 1 and Object.CenterY <= 1 Then  
    Object.Bottom = 2  
    Object.Left = 2  
End If
```

## Related Topics

### Description

[Working with Objects](#)

## Resizing Objects

Related

You can resize objects using the [StretchType](#), [Height](#), and [Width](#) properties in the Object object. You use the StretchType property to specify the type of stretching behavior assigned to an object. Set the value to 0 for normal behavior (the anchor is the center of the object and opposite sides both move, as when you stretch normally in ABC). Set the value to 1 for fixed side behavior in ABC (the anchor is the opposite side from the handle grabbed, as when you hold down the **Ctrl** key and stretch in ABC). If you set the value to 1 and then resize the object with ABC OLE Automation statements, the top and left sides are fixed (as if you were stretching from the right or bottom center handle).

For example, the following statements set the [StretchType property](#) to fix the top and left sides and then set the height to 1.25 inches and the width to 2.5 inches.

```
Object.StretchType = 1  
Object.Height = 1.25  
Object.Width = 2.5
```

## Related Topics

### Description

[Working with Objects](#)

## Changing the Display Order of Objects

Related

You can change the order in which objects display using the [ToBack](#) and [ToFront](#) methods. The methods are in both the Chart object and the Object object.

The [ToBack method](#) with an object is equivalent to clicking the To Back button in the ABC Selection ribbon with one or more objects selected.

The [ToFront method](#) with an object is equivalent to clicking the To Front button in the ABC Selection ribbon with one or more objects selected.

The first statement below moves the selected objects to the back. The second statement moves the object to the back.

```
Chart.ToBack  
Object.ToBack
```

The first statement below moves the selected objects to the front. The second statement moves the object to the front.

```
Chart.ToFront  
Object.ToFront
```

## Related Topics

### Description

[Working with Objects](#)

## Setting the Current Drawing Position

Related

When you draw using ABC, you click where you want to place the next object, text, or line. With ABC OLE Automation, the position is determined by the most recent draw position (plus [DrawSpacingX](#) and [DrawSpacingY](#) for shapes; see [Drawing Shapes](#) for more information). You can specify a location using the [DrawPositionX](#) and [DrawPositionY](#) properties in the Chart object. The position you specify is used for the next object drawn or the next object pasted or pasted special (if those methods do not specify a different position).

For example, the following statements specify that the center of the next object is to be four inches from the left side of the page and five inches from the top of the page.

```
Chart.DrawPositionX = 4  
Chart.DrawPositionY = 5
```

## Related Topics

### Description

[Working with Objects](#)

## Cutting, Copying, Pasting, Duplicating, and Clearing Objects

Related

[Cutting, Copying, and Pasting Objects](#)

[Using Special Clipboard Formats](#)

[Duplicating Objects](#)

[Clearing Selected Objects](#)

[Resizing Objects](#)

## Related Topics

### Description

[Working with Objects](#)

## Cutting, Copying, and Pasting Objects

Related

You use the [Cut](#), [Copy](#), and [Paste](#) methods in the Chart object just as you would open the ABC Edit menu and choose the Cut, Copy, and Paste command. All three methods return a Boolean value equal to True if the operation was successful or False if the operation failed.

The [Paste method](#) has the additional ability to specify where to place the pasted object.

**Paste**([*HorizontalLocation*] [, *VerticalLocation*])

For example, the following statements cut whatever is currently selected and paste it so its upper left corner is two inches from the left margin and three inches from the top margin of the page.

```
Successful = Chart.Cut
If Successful Then
    Successful = Chart.Paste(2,3)      ' Paste if cut is successful
End If
If Not Successful Then
    MsgBox "Cut or Paste Failed", 48  ' Deal with cut or paste failure
                                     ' 48 is the exclamation point
End If
```

If you omit the location for the [Paste method](#), it places the object as described in [Setting the Current Drawing Position](#).

## Related Topics

### Description

[Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#)

## Using Special Clipboard Formats

Related

The [PasteSpecial method](#) in the Chart object lets you paste from the Clipboard specifying a format. It is the same as choosing the Paste Special command in the ABC Edit menu and then specifying the format to use for the paste. The method returns a Boolean value equal to True if the operation was successful or False if the operation failed.

**PasteSpecial** (*Format* [, *AsIcon*] [, *HorizontalLocation*] [, *VerticalLocation*])

You also can specify that the object on the Clipboard be pasted as an icon using the second parameter. This is equivalent to selecting the Display As Icon option in the ABC Paste Special dialog box. You can specify the location of the paste. If you omit the location for the [PasteSpecial method](#), it places the object as described in [Setting the Current Drawing Position](#).

The following table shows the formats for the [PasteSpecial method](#) and for the [ClipboardFormatAvailable property](#), which is explained below.

Value	Format
0	ABC Native
1	OLE Client Embed
2	ABC Rich Text
3	Rich Text Format (RTF)
4	Unformatted text
5	Metafile
6	Device Independent Bitmap
7	Bitmap
8	OLE Client Link

For example, the following statement pastes the Clipboard object as an OLE client link icon with its upper left corner two inches from the left margin and three inches from the top margin of the page.

```
Successful = Chart.PasteSpecial(8,True,2,3)
```

You use the [ClipboardFormatAvailable property](#) in the Chart object to find out whether the object in the Clipboard is in a format that you want. The property returns a Boolean value equal to True if the format is available or False if the format is not available.

The [ClipboardFormatAvailable property](#) uses the same values and formats as the [PasteSpecial method](#) described above.

For example, the following puts the Boolean value True or False in CanPaste depending on whether the object currently in the Clipboard can be pasted as a DIB.

```
CanPaste = Chart.ClipboardFormatAvailable(6)
```

## Related Topics

### Description

[Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#)

## Duplicating Objects

Related

The [Duplicate method](#) is in both the Chart object and the Object object. The method in the Chart object duplicates whatever is currently selected and returns a Boolean value equal to True if the operation was successful or False if the operation failed.

In the Object object, the [Duplicate method](#) makes a duplicate of that object and returns the duplicate object.

For example, the following statements duplicate the selected chart objects, then make a duplicate of the object stored in the variable Object.

```
Successful = Chart.Duplicate  
If Successful Then  
    DuplicatedObject = Object.Duplicate  
End If
```

## Related Topics

### Description

[Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#)

## Clearing Selected Objects

Related

The [Clear\\_method](#) of the Chart object deletes the selected objects from the chart. It is the equivalent of pressing the **Del** key or opening the ABC Edit menu and choosing Clear. The method deletes whatever is currently selected and returns a Boolean value equal to True if the operation was successful or False if the operation failed. For example, the following statement deletes the selected objects.

```
Successful = Chart.Clear_
```

The [Clear\\_method](#) of the Object object deletes the object object. You usually use it as part of a routine using the [SetDefaults\\_method](#). The Clear\_ method deletes the indicated object and returns a Boolean value equal to True if the operation was successful or False if the operation failed. For example, the following statement deletes the indicated object.

```
Successful = Chart.DefaultObject.Clear_
```

## Related Topics

### Description

[Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#)

## Restoring Objects



The [RestorePicture method](#) of the Object object lets you restore bitmap and OLE client objects to their original size. This method is nearly the same as the OLE object's [RestorePicture method](#). The difference is that this method handles bitmaps as well as OLE objects, while the OLE object's [RestorePicture method](#) only works on OLE objects.

For example, the following statement restores an object.

```
PasteObject.RestorePicture
```

## Related Topics

### Description

[Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#)

## Using OLE Client Objects Related

You can use ABC OLE Automation to work with objects that are linked or embedded in a chart using OLE. To work with linked or imbedded objects, you use the [InsertObjectFromFile method](#), the [PasteLink method](#), the [UpdateFields method](#), the [OLE property](#), the [ObjectType property](#), and the [DoVerb method](#).

You use the [InsertObjectFromFile method](#) of the Chart object to insert a new OLE client object from a file. You can optionally add parameters to specify that the file be inserted as an icon or linked. The method returns the file that is inserted as an object. The [InsertObjectFromFile method](#) is equivalent to opening the ABC Edit menu, choosing Insert Object, choosing the Create from File option, selecting the file you want to insert, and clicking OK. The *AsIcon* element is equivalent to selecting the Display As Icon option. The *AsLink* element is equivalent to selecting the Link to File option.

**InsertObjectFromFile**(*Filename* [, *AsIcon*] [, *AsLink*])

For example, the following statement inserts an Excel file into the chart and sets *InsertedOleObject* equal to the new object.

```
InsertedOleObject = Chart.InsertObjectFromFile(C:\EXCEL\DATA.XLS)
```

You use the [PasteLink method](#) of the Chart object to paste the contents of the Clipboard into the chart and link the file that is the source of the contents of the chart. The [PasteLink method](#) is equivalent to opening the ABC Edit menu and choosing Paste Link. For example, the following statement pastes and links the contents of the Clipboard 2 inches from the left side of the chart and 3 inches from the top of the chart.

```
Chart.PasteLink(2,3)
```

You use the [UpdateFields method](#) of the Chart object to update all the linked fields in the chart. For example, the following statement updates the linked fields in the chart.

```
Chart.UpdateFields
```

The [UpdateFields method](#) is equivalent to opening the ABC Edit menu, choosing Links, and choosing Update Now.

You use the [OLE property](#) of the Object object and the [ObjectType property](#) of the Object object to find the short object class name of an object that is embedded or linked.

For example, these statements append the name of the OLE object type to the end of the object's text for all linked objects.

```
Set ABCObjects = Chart.Objects
Do
  Set Object = ABCObjects.ItemFromAll
  If Object.OLE.ObjectType <> "" Then
    Object.Text = Object.Text + "OLE: " + OLE.ObjectType
  End If
Loop While Object.Valid
```

**Note:** You cannot change the value in the [ObjectType property](#).

You use the [DoVerb method](#) of the OLE object to specify an OLE verb to execute if the object is a linked or embedded OLE object. If you do not specify a verb, the default verb is used.

For example, the following statements find the OLE objects in a chart and execute the default verb for each of them.

```
Set ABCObjects = Chart.Objects
Do
  Set Object = ABCObjects.ItemFromAll
  If Object.OLE.ObjectType <> "" Then
    Object.OLE.DoVerb
  End If
Loop While Object.Valid
```

### Restoring OLE Objects

## Related Topics

### Description

[Working with Objects](#)

## Restoring OLE Objects

Related

The [RestorePicture method](#) of the OLE object lets you restore OLE client objects to their original size. This method is nearly the same as the Object object's [RestorePicture method](#). The difference is that this method only works on OLE objects, while the Object object's [RestorePicture method](#) handles bitmaps as well as OLE objects.

For example, the following statement restores an OLE object.

```
OLEObject.RestorePicture
```

## Related Topics

### Description

[Using OLE Client Objects](#)

## Speeding Actions

Related

One way to speed the actions of ABC OLE Automation when there is a series of actions is to omit drawing each action by using the [NoRepaint property](#) in the Chart object. You then update the screen using the [Repaint method](#) in the Chart object. This can provide a 15% to 20% increase in speed.

For example, the following statements turn off painting, create 200 shapes, and then repaint the chart.

```
Chart.NoRepaint = True
For DrawFast = 1 to 200
    Chart.DrawShape
Next DrawFast
Chart.NoRepaint = False
Chart.Repaint
```

You can also speed actions when you are creating many objects that all have the same characteristics. For example, suppose you are creating many shapes and want them all to have text that is red 17-point Futura with a green shadow. Create one object, set the style for it, and then use the [Duplicate method](#) of the Object object to make as many copies as you need.

You can also speed actions by setting the defaults for Line\_, Shape, and TextBlock objects using the [SetDefaults method](#). With that method, you create an object with the defaults you want and then pass it to the method. For example, the following statement sets the defaults for lines to the defaults of the line object named LineObject.

```
Chart.SetDefaults LineObject
```

## Related Topics

### Description

[Working with Objects](#)

[Setting Defaults](#)

## Undoing Actions

**Related**

You can choose ABC's Undo command using the [Undo method](#) of the Application object. You can find out if there is anything to undo using the [UndoAvailable property](#) of the Application object. Using the [Undo method](#) is equivalent to opening the ABC Edit menu and choosing Undo.

The following statements undo the last action if it is available. Whether the last action was undone is put in the status bar.

```
If ABC.UndoAvailable = True Then
    ABC.Undo
    ABC.StatusBar = "Last action undone."
Else
    ABC.StatusBar = "Nothing available to undo."
End If
```

## Related Topics

### Description

[Working with Objects](#)

Related

## Working with Shapes

Related

[Drawing Shapes](#)

[Using the Shape Palette](#)

[Adding Text to Shapes](#)

[Working with Shape Numbers](#)

[Formatting Shapes](#)

[Replacing Shapes](#)

[Selecting Shapes](#)

[Working with Notes](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Drawing Shapes Related

You can draw any shape in the Shape Palette. Use the [DrawShape method](#) of the Chart object to draw shapes. By default, [DrawShape](#) uses the current shape in the Shape Palette. For information on specifying the shape, see the [Choosing a Shape in the Palette](#).

```
Set ABCObject = Chart.DrawShape
```

All the shape palettes that ship with ABC have predefined names that appear in the hintline when the mouse pauses over them, and which are listed in the documentation that ships with ABC. In ABC the shape's name is defined in the Shape Properties dialog box. You can open the Shape Properties dialog box by choosing Shape Properties in the Palette menu of the Shape Palette.

In ABC OLE Automation you can optionally specify the type of shape you want to draw by specifying the shape's name. The program uses a "loose matching" routine so, for example, setting the shape's name to "Proc" chooses "Process." If more than one shape matches the value you set, the exact match is used first. If there is not exact match, the first one alphabetically is used.

```
Set ABCObject = Chart.DrawShape("Proc")
```

Shapes are automatically placed at the chart's current drawing position. (See [Setting the Current Drawing Position](#) for more information on the current drawing position.) Alternatively, you can use the [DrawDirection property](#) of the Chart object to specify the direction for placing new shapes. The following table shows the values for the [DrawDirection property](#).

Value	Description
0	North
1	East
2	South
3	West
10	Stacked

You specify the horizontal and vertical distance from a shape to the next one you create using the [DrawSpacingX property](#) and [DrawSpacingY property](#) of the Chart object. The [DrawSpacingX property](#) and [DrawSpacingY property](#) are equivalent to choosing Preferences in the File menu, clicking the Alignment Options button, and entering horizontal and vertical spacing.

You can use the [NextShapeHeight property](#) and [NextShapeWidth property](#) of the Chart object to specify the height of the next shape to be drawn.

For example, the following statements set the horizontal spacing to two inches, the vertical spacing to three inches. They then specify the height for the next shape drawn and then that the next shape should be to the right of the current shape.

```
Chart.DrawSpacingX = 2           ' Horizontal spacing 2"  
Chart.DrawSpacingY = 3           ' Vertical spacing 2"  
Chart.NextShapeHeight = .5       ' Height of next shape .5  
Chart.NextShapeWidth = .5        ' Width of next shape .5  
Chart.DrawDirection = 1          ' Next shape toward right
```

See the [Moving Objects](#) and [Resizing Objects](#) for information on changing the size and position of shapes.

See [Drawing Lines that Connect Shapes](#) for information on drawing lines to connect shapes.

## Related Topics

### Description

[Working with Shapes](#)

## Using the Shape Palette

Related

ABC provides a wide variety of shape palettes you can use in drawing charts. Each palette contains several shapes that you can choose from. With ABC OLE Automation, you can display or hide the Shape Palette, open a different Shape Palette, and choose a shape from the palette.

[Displaying and Hiding the Shape Palette](#)

[Opening a Different Shape Palette](#)

[Choosing a Shape in the Palette](#)

## Related Topics

### Description

[Working with Shapes](#)

## Displaying and Hiding the Shape Palette

Related

Use the [ShapePaletteVisible property](#) of the Application object to display or hide the Shape Palette or to determine whether the Shape Palette is visible. When this property is True, the Shape Palette is displayed; when False, it is hidden.

```
ABC.ShapePaletteVisible = True ' Displays the Shape Palette
```

## Related Topics

### Description

[Using the Shape Palette](#)

## Opening a Different Shape Palette

Related

You can open any of the Shape Palettes that ship with ABC. Use the [CurrentShapePalette property](#) of the Chart object to open a different Shape Palette or determine the name of the current Shape Palette. The name of the Shape Palette appears in the title bar of the palette. The name is not related to the filename of the palette.

For example, the following statement opens the Auditing Shape Palette.

```
Chart.CurrentShapePalette = "Auditing"
```

## Related Topics

### Description

[Using the Shape Palette](#)

## Choosing a Shape in the Palette

Related

The [DrawShape method](#) of the Chart object draws the current shape in the Shape Palette, unless you specify a shape to draw with the method. In ABC you choose a shape to be the current shape by clicking it in the Shape Palette. The name of the shape appears in the hintline when you pass over the shape with the mouse.

With ABC OLE Automation you use the [CurrentShape property](#) of the Chart object to choose a shape as the current shape. The program uses a "loose matching" routine so, for example, setting the [CurrentShape property](#) to "Dec" chooses "Decision." If more than one shape matches the value you set, the exact match is used first. If there is not exact match, the first one alphabetically is used.

For example, the following statements choose the Decision shape as the next shape to be drawn, and then draw the shape.

```
Chart.CurrentShape = "Decision"           ' Decision is current shape
Set ABCObject = Chart.DrawShape          ' Draw the current shape
```

## Related Topics

### Description

[Using the Shape Palette](#)

## Adding Text to Shapes

Related

You can use ABC OLE Automation to add text inside any shape. The text appears inside the text area defined for the shape. Adding text to a shape is equivalent to typing while a shape is selected in ABC.

To add text to a shape, use the [Text property](#) of the Object object. The following example draws the shape shown above.

```
Dim ABCObject As Object  
  
Set ABCObject = Chart.DrawShape("External Operation")  
ABCObject.Text = "Text inside a shape"
```

If you are reading the text from a shape, you can use the [TextLF property](#) to preserve the Returns. If you use the [Text property](#), the Returns are changed to spaces.

```
ShapeText = Shape1.TextLF
```

## Related Topics

### Description

[Working with Shapes](#)

[Fitting Shapes to Text](#)

## Fitting Shapes to Text

Related

You can automatically fit shapes to the size of the text inside them. This is especially useful when the length of the text string may vary.

In ABC, you do this by clicking the Text tool and then the Fit Shape button in the ribbon when the shape is selected. With ABC OLE Automation, you use the [FitShapeToText method](#) of the Shape object.

The following example draws a shape, adds text to the shape, then fits the shape to the text.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape("Document")
ABCObject.Text = "This is a sample of fitting shapes to text."
ABCObject.Shape.FitShapeToText
```

## Related Topics

### Description

[Adding Text to Shapes](#)

## Working with Shape Numbers

Related

With ABC OLE Automation you can read a shape's number, renumber shapes, format the shape numbers, and hide shape numbers.

[Numbering Shapes](#)

[Formatting Shape Numbers](#)

[Hiding Shape Numbers](#)

## Related Topics

### Description

[Working with Shapes](#)

## Numbering Shapes

Related

You can use various numbering systems for shapes, such as *1, 2, 3; 1.1, 1.2, 1.3*; or even text strings.

The number used for the next new shape you draw is stored in the [NextNumber property](#) of the Chart object. The number is kept as a text string, since the number can contain text as well as numbers.

The [NextNumber property](#) is incremented automatically each time you draw a shape. If [NextNumber](#) contains text with a number, the text remains and the number is incremented. For example, "Step 5" will become "Step 6" when a new shape is drawn. If [NextNumber](#) contains only text, the text remains without incrementing. For example, "Step Five" will stay as "Step Five" even after a new shape is drawn. This is especially useful when you want the shape number to be a placeholder for a company name or department name.

The [Number property](#) of the Shape object contains the shape number for a specific shape. When you draw a shape, the value in the chart's [NextNumber](#) is stored in [Number](#), and [NextNumber](#) is incremented. You can change a shape's number by changing the value of the shape's [Number property](#).

You also can use the [Renumber method](#) to change a shape's number. [Renumber](#) changes a shape's [Number property](#) to the chart's [NextNumber](#) value, and increments [NextNumber](#).

The following example illustrates [NextNumber](#) and [Renumber](#).

```
Sub ShapeNumbers()  
    Dim ShapeOne As Object, ShapeTwo As Object, ShapeThree As Object  
    Dim Chart As Object, ABC As Object  
    Set ABC = CreateObject("ABCFlow.Application")  
    Set Chart = ABC.ActiveChart  
  
    Set ShapeOne = Chart.DrawShape  
    Set ShapeTwo = Chart.DrawShape  
  
    ShapeOne.Shape.Number = "Step 1"  
    Chart.NextNumber = "Step 2"  
    ShapeTwo.Shape.Renumber  
    Set ShapeThree = Chart.DrawShape  
End Sub  
  
' NextNumber initially defined as 1  
' ShapeOne.Number=1; NextNumber=2  
' ShapeTwo.Number=2; NextNumber=3  
  
' ShapeOne.Number=Step 1; NextNumber=3  
' NextNumber=Step 2  
' ShapeTwo.Number=Step 2; NextNumber=Step 3  
' ShapeThree.Number=Step 3; NextNumber=Step 4
```

## Related Topics

### Description

[Working with Shape Numbers](#)

## Formatting Shape Numbers

Related

You can format shape numbers by choosing a typeface, size, color, and text attributes.

In ABC you format shape numbers by choosing Preferences in the File menu to open the Preferences dialog box. Select the Indicator Options button at the side of the dialog box, and choose number formatting options in the Number Font area of the dialog box.

Using ABC OLE Automation, you can select the same options using the NumberFont object of the Chart object. Like other font objects, the NumberFont object has the following properties.

Property	Description
<a href="#">Bold</a>	True if text is <b>bold</b> ; False if text is not bold.
<a href="#">Color</a>	The color used in shape numbers. This value can be one of the 16 color constants, such as ABC.Blue.
<a href="#">Italic</a>	True if text is <i>italic</i> ; False if text is not italic.
<a href="#">Name</a>	The typeface name used for shape numbers, such as "Arial" or "Roman."
<a href="#">Size</a>	The point size of shape numbers.
<a href="#">Strikethrough</a>	True if text is <del>strikethrough</del> ; False if text is not strikethrough. This attribute is not available to shape numbers in ABC.
<a href="#">Underline</a>	True if text is <u>underline</u> ; False if text is not underline.

See [Formatting Text](#) for more information on these properties.

The following statements change shape number text to Helvetica 12-point bold italic.

```
Chart.NumberFont.Name = Helvetica
Chart.NumberFont.Size = 12
Chart.NumberFont.Bold = True
Chart.NumberFont.Italic = True
```

The [Opaque property](#) is not available in the NumberFont object. Shape numbers are opaque when the other text in the shape is opaque, and transparent when the other text is transparent. For example, the following statement makes both shape text and shape numbers opaque.

```
ABCObject.Font.Opaque = True
```

## Related Topics

### Description

[Working with Shape Numbers](#)

## Hiding Shape Numbers

Related

Just as with ABC, shape numbers automatically appear within shapes, but you can hide the shape numbers if you wish. This feature is useful when you do not want numbers to appear in certain shape types, such as documents or decisions.

In ABC you hide shape numbers in selected shapes by clicking the Number tool in the toolbox and deselecting the Show Number button in the ribbon. Select the Number Button to display shape numbers for selected shapes.

With ABC OLE Automation, you use the [NumberShown property](#) of the Shape object to display or hide shape numbers. Make [NumberShown](#) equal to True to display numbers, as on the first statement below or False to hide numbers, as in the second statement below.

```
ABCObject.Shape.NumberShown = True  
ABCObject.Shape.NumberShown = False
```

## Related Topics

### Description

[Working with Shape Numbers](#)

## Formatting Shapes

**Related**

You can format shapes by changing their color, patterns and styles, border width, and shadow offset.

[Fill, Border, and Shadow Colors](#)

[Fill Pattern](#)

[Border Style and Width](#)

[Shadow Style and Width](#)

## Related Topics

### Description

[Working with Shapes](#)

## Fill, Border, and Shadow Colors

Related

You can color a shape by setting its fill color, its border color, and its shadow color.

You set the fill color for shapes using the [FillColor property](#) of the Shape object or the [Color property](#) of the Object object. Both properties produce the same effect.

For example, the following statements draw a shape and then change its fill color to blue using the [FillColor property](#). They then change its fill color to red using the [Color property](#).

```
Dim NewObj1 As Object
Set NewObj1 = Chart.DrawShape
NewObj1.Shape.FillColor = ABC.BLUE
NewObj1.Color = ABC.RED
```

You set the border color for shapes using the [BorderColor property](#) of the Shape object. For example, the following statement makes the border of a shape blue.

```
NewObj1.Shape.BorderColor = ABC.BLUE
```

You set the shadow color for shapes using the [ShadowColor property](#) of the Shape object. For example, the following statement makes the shadow of a shape blue.

```
NewObj1.Shape.ShadowColor = ABC.BLUE
```

## Related Topics

### Description

[Formatting Shapes](#)

## Fill Pattern



You can fill a shape with any of the patterns available in ABC.

In ABC you change a selected shape's fill pattern by clicking the Shape tool in the toolbox, and then choosing a pattern from the Fill Pattern button in the ribbon.

To set or read a shape's fill pattern with ABC OLE Automation, use the [FillPattern property](#) of the Shape object. Set [FillPattern](#) to 0 for a transparent fill or to 1 for a solid fill. See [FillPattern](#) for each available pattern.

The following statements draw a shape and then change its fill pattern to vertical stripes.

```
Set ABCObject = Chart.DrawShape("Process")
ABCObject.Shape.FillPattern = 4
```

## Related Topics

### Description

[Formatting Shapes](#)

## Border Style and Width

Related

You can choose different line styles for shape borders. A shape border includes not only the outside edge of a shape, but also any interior lines used in the shape (for example, the concentric circles on the inside of a 5 1/2" floppy disk shape). ABC provides many useful border styles, including solid and dashed lines and an invisible border.

In ABC you change a selected shape's border by clicking the Shape tool in the toolbox, and then choosing a style from the Border Style button in the ribbon. You set the width of the border by clicking the Apply button beside the Border Style and then choosing a value in the Width box.

To set or read a shape's border style with ABC OLE Automation, use the [BorderStyle property](#) of the Shape object. Set [BorderStyle](#) to 0 for an invisible border and 1 for a solid line border. See [BorderStyle](#) for each available style.

Use the [BorderWidth property](#) of the Shape object to change or read the width of the border. [BorderWidth](#) can have a value ranging from 1 (hairline) to 5 (thickest). **Note:** [BorderWidth](#) is applied only if [BorderStyle](#) is 1; it does not apply to dashed or dotted borders.

The following statements draw a Process shape and then change its border to a dotted line.

```
Set ABCObject = Chart.DrawShape("Process")
ABCObject.Shape.BorderStyle = 3
```

The following statements draw a decision shape and change its border to a very thick solid line.

```
Set ABCObject = Chart.DrawShape("Process")
ABCObject.Shape.BorderStyle = 1
ABCObject.Shape.BorderWidth = 5
```

## Related Topics

### Description

[Formatting Shapes](#)

## Shadow Style and Width Related

You can add a drop shadow to shapes and choose the position and width of the shadow.

In ABC you add a shadow to a selected shape by clicking the Shape tool in the toolbox, and then choosing a style from the Shadow button in the ribbon. You set the width of the shadow by clicking the Apply button beside the Shadow Style and then choosing a value in the Width box.

With ABC OLE Automation you add a shadow using the [ShadowStyle property](#) of the Shape object. [ShadowStyle](#) can have a value from 0 to 4, with 0 being no shadow and 1 through 4 being the positions shown in [ShadowStyle](#)

The width of a shadow (the distance the shadow appears away from the shape) is determined by the [ShadowOffset property](#) of the Shape object. [ShadowOffset](#) can have a value ranging from 1 (hairline) to 5 (thickest).

You also can use the [ShadowStyle](#) and [ShadowOffset](#) properties to read the values of the current shadow of a shape.

The following statements draw a shape, and then add a drop shadow with medium thickness.

```
Set ABCObject = Chart.DrawShape("Document")
ABCObject.Shape.ShadowStyle = 2
ABCObject.Shape.ShadowOffset = 3
```

## Related Topics

### Description

[Formatting Shapes](#)

## Replacing Shapes

**Related**

You can replace one or more shapes in a chart with a different type of shape. When you replace shapes, the new shapes connect to the lines of the old shapes.

In ABC you replace selected shapes by choosing the new shape in the Shape Palette, clicking the Shape tool in the toolbox, and clicking the Replace Shape button in the ribbon.

With ABC OLE Automation, use the [ReplaceShape method](#) of the Shape object to replace shapes. You can replace shapes with the chart's current shape or with any shape type you specify.

*ShapeObject*.**ReplaceShape** [*ShapeType*]

*ShapeType* is an optional parameter that specifies the shape type that will be used to replace the shape referred to in *ShapeObject*.

The following example replaces all Operation shapes in a chart with External Operation shapes.

```
Set Obj = ABC.ActiveChart.Objects
Do
  Set ABCObject = Obj.ItemFromShapes
  If ABCObject.Shape.ShapeName = "Operation" Then
    ABCObject.Shape.ReplaceShape "External Operation"
  End If
While ABCObject.Valid
```

## Related Topics

### Description

[Working with Shapes](#)

## Selecting Shapes

Related

ABC OLE Automation provides several ways to select shapes. You can select a single shape, all shapes of a particular type, or all shapes in the chart.

To select a single shape, use the [Selected property](#) of the Object object. Set the [Selected property](#) to True to select the shape or False to deselect the shape. For example, the following statements draw a shape and then select it.

```
Dim ABCShape As Object
Set ABCShape = Chart.DrawShape
ABCShape.Selected = True
```

To select all shapes of a particular type, such as Process or Decision, use the [SelectShapeType method](#) of the Chart object. This method takes one parameter: a string indicating the type of shape. The statement below selects all Document shapes.

```
Chart.SelectShapeType "Document"
```

To select all the shapes in a chart, use the [Select method](#) of the Chart object. The [Select method](#) can select all shapes, all lines, or all objects in a chart. It can also be used to deselect all objects. The [Select method](#) takes one parameter, an integer indicating the selection.

Select Value	Result
0	Selects all shapes
1	Selects all lines
2	Selects all objects (shapes, lines, text blocks)
3	Deselects all objects

For example, the following statements deselect all objects in a chart and then select only the lines.

```
Chart.Select 3
Chart.Select 1
```

### [Deselecting Shapes](#)

## Related Topics

### Description

[Working with Shapes](#)

## Deselecting Shapes

Related

ABC OLE Automation provides three ways to deselect shapes.

You can deselect a single object by using the selected property of the Object object. Set the [Selected property](#) to False to deselect the shape.

You can deselect all the objects that are currently selected by using the [DeselectAll method](#) of the Chart object. The [DeselectAll method](#) requires no extra parameters.

You can deselect all the objects that are currently selected by using the [Select method](#) of the Chart object, but the [Select method](#) must be followed by the number 3 to deselect objects.

In the statements below, the first line deselects only the selected object. The last two lines deselect all objects.

```
ABCObject.Selected = False  
Chart.DeselectAll  
AllChart.Select 3
```

## Related Topics

### Description

[Selecting Shapes](#)

## Working with Notes

Related

[Opening the Note Window](#)  
[Attaching a Note to a Shape](#)  
[Choosing Note Indicators](#)  
[Formatting Note Text](#)  
[Printing Notes](#)

## Related Topics

### Description

[Working with Shapes](#)

## Opening the Note Window

Related

The Note window displays notes for the currently selected shape.

In ABC you open and close the Note window by clicking the Note button at the bottom of the ABC window.

Using ABC OLE Automation, you open and close the Note window using the [NoteViewerVisible property](#) of the Application object. Set this property to True to open the Note window or False to close the Note window. You also can use this property to check whether the window is already open.

The following example checks to see if the Note window is open, and then closes the Note window.

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.Application")
If ABC.NoteViewerVisible Then      ' If Note window is open
    ABC.NoteViewerVisible = False  ' Close the Note window
End If
```

## Related Topics

### Description

[Working with Notes](#)

## Attaching a Note to a Shape

Related

You can attach notes to any shape in a chart. In ABC, notes are added in the Note window while the shape is selected. With ABC OLE Automation, you do not need to open the Note window to attach notes to a shape.

Use the [NoteText property](#) of the Shape object to attach notes to shapes.

The following example draws a shape, then adds a note to the shape.

```
Dim ShapeObject As Object

Set ShapeObject = Chart.DrawShape("Document")
ShapeObject.Text = "Text inside a shape"
ShapeObject.Shape.NoteText = "This is note text attached to the shape"
```

If you are reading the note text from a shape, you can use the [NoteTextLF property](#) to preserve the Returns. If you use the [NoteText property](#), the Returns are changed to spaces.

```
NoteText = Shape1.NoteTextLF
```

## Related Topics

### Description

[Working with Notes](#)

## Choosing Note Indicators Related

You can use indicators to identify shapes that have attached notes. Indicators include shadows around the symbol and symbols next to the shape number. The default indicator for notes is **-N**. You can use up to three characters to create your own indicators.

The text settings used for indicator symbols are the same as for the shape number. See the [Formatting Shape Numbers](#) for more information on formatting shape numbers.

In ABC you choose note indicators in the Indicator Options panel of the Preferences dialog box.

With ABC OLE Automation you use the [NoteIndicator](#) and [NoteShadow](#) properties of the Chart object. The [NoteIndicator property](#) identifies the three character symbol as a string. The [NoteShadow property](#) is a Boolean value that determines whether a shadow displays around shapes with attached notes.

```
Chart.NoteIndicator = "*N*"           ' Set *N* as the note symbol
Chart.NoteShadow = 1                 ' Use shadow to indicate notes
```

## Related Topics

### Description

[Working with Notes](#)

## Formatting Note Text

**Related**

You can format note text just as you can format other text objects. The formatting for note text appears in the Note window and in printed notes. Note text is formatted for each shape individually.

The [NoteFont property](#) of the Shape object returns a font object with properties that you can set. That object has the following properties.

Property	Description
<a href="#">Bold</a>	True if text is <b>bold</b> ; False if text is not bold
<a href="#">Italic</a>	True if text is <i>italic</i> ; False if text is not italic
<a href="#">Strikethrough</a>	True if text is <del>strikethrough</del> ; False if text is not strikethrough
<a href="#">Underline</a>	True if text is <u>underline</u> ; False if text is not underline
<a href="#">Name</a>	The typeface name of the font
<a href="#">Size</a>	The point size of the font

See [Formatting Text](#) for more information on these properties.

As with other text objects, the [NoteFont property](#) of the shape object returns a font object that has the [Opaque property](#), but it is not useful for formatting notes. The following example formats the note text for all shapes in a chart.

```
Dim ABC As Object
Dim AllShapes As Object
Dim CurrentShape As Object

Set ABC = CreateObject("ABCFlow.Application")
Set AllShapes = ABC.ActiveChart.Objects

Do
    Set CurrentShape = AllShapes.ItemFromShapes
    CurrentShape.Shape.NoteFont.Name = "Arial"
    CurrentShape.Shape.NoteFont.Size = 12
    CurrentShape.Shape.NoteFont.Italic = True
Loop While CurrentShape.Valid
```

## Related Topics

### Description

[Working with Notes](#)

## Printing Notes

Related

You can print the notes that are attached to shapes.

In ABC there are two ways to print notes. Choose the Print command directly from the Note window, or open the File menu in the main window and choose Print, and then select the Print Notes option in the Print dialog box.

Using ABC OLE Automation, you print notes in a way similar to using the Print dialog box. The [PrintOut](#) and [PrintSelected](#) methods print a chart, and can also print notes associated with the chart.

**PrintOut** [*FromPage*] [, *ToPage*] [, *Copies*] [, *FitToPage*] [, *PrintNotes*]

**PrintSelected** [*Copies*] [, *FitToPage*] [, *PrintNotes*]

All parameters in these methods are optional. To print notes, the *PrintNotes* parameter must be 1 (True). Look at the following examples.

```
PrintOut ( , , 2, , 1)      ' Print 2 copies of chart and notes
PrintSelected ( , 1, 1)    ' Print selected objects to fill the page with attached notes
```

## Related Topics

### Description

[Working with Notes](#)

Related

## Working with Lines

Related

[Drawing Lines](#)

[Connecting Existing Lines to Shapes](#)

[Setting Line Routing](#)

[Formatting Lines](#)

[Displaying Nodes on Connecting Lines](#)

[Setting Line Crossovers](#)

[Attaching Text to Lines](#)

[Deleting Lines](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Drawing Lines

Related

You can draw lines by specifying the starting and ending points of a line in space, by drawing that is connected to one shape, or by connecting two shapes with a line. You access information about lines using the [Line\\_property](#) of the Object object.

[Drawing Unconnected Lines](#)

[Drawing Lines to One Shape](#)

[Drawing Lines that Connect Shapes](#)

## Related Topics

### Description

[Working with Lines](#)

## Drawing Unconnected Lines

Related

Unconnected lines are lines that are drawn from one point to another and are not connected to any shapes or lines.

Use the [DrawFreeLine method](#) of the Chart object to draw an unconnected line.

**DrawFreeLine** (*XPosition*, *YPosition*)

The line starts at the chart's current drawing position and ends at the point you specify with *XPosition* and *YPosition*. The X and Y positions are measured from the top left corner of the ABC page. By default, the positions are measured in inches, but you can measure position in centimeters by changing the [Units property](#) of the Preferences object. See [Setting the Current Drawing Position](#) for more information.

For example, the following statement draws a line from (1,1) to (2,4).

```
Chart.DrawPositionX = 1
Chart.DrawPositionY = 1
Set NewLine = Chart.DrawFreeLine (2, 4)
```

The type of routing used for the line is determined by the chart's current line routing. See [Setting Line Routing](#) for more information.

## Related Topics

### Description

[Drawing Lines](#)

## Drawing Lines to One Shape

Related

You can draw lines that are unconnected on one end, and connected to a shape on the other end. If you move the shape, the connected line follows.

Use the [DrawLineToOneObject method](#) of the Chart object to draw lines that connect to only one shape.

**DrawLineToOneObject** (*ShapeObject* [,*EnterDirection*])

The line starts at the chart's current drawing position and ends at the shape you specify with *ShapeObject*. See the [Setting the Current Drawing Position](#) for information on setting the chart's current drawing position.

You can optionally use a second parameter that specifies the direction that the line enters the shape. The following table describes each of the *EnterDirection* values.

Value	Direction
0	North
1	East
2	South
3	West

For example, the following statement draws a line from the current position to the shape specified in *ShapeObject*.

```
Set NewLine = Chart.DrawLineToOneObject (ShapeObject, 0)
```

The type of routing used for the line is determined by the chart's current line routing. See [Setting Line Routing](#) for more information.

## Related Topics

### Description

[Drawing Lines](#)

## Drawing Lines that Connect Shapes

Related

You can draw lines that connect two shapes. If you move either of the shapes, the connected line follows.

Use the [DrawLine method](#) of the Chart object to draw lines that connect two shapes.

**DrawLine** (*ShapeObject1*, *ShapeObject2* [,*ExitDirection*] [,*EnterDirection*] )

The line starts at *ShapeObject1* and ends at *ShapeObject2*. You can optionally use third and fourth parameters that specify which direction the line exits *ShapeObject1* and enters *ShapeObject2*. The following chart describes each of the direction values.

Value	Direction
0	North
1	East
2	South
3	West

For example, the following statement draws a line from the bottom of Shape1 to the top of Shape2.

```
Set NewLine = Chart.DrawLine (Shape1, Shape2, 2, 0)
```

The type of routing used for the line is determined by the chart's current line routing. See [Setting Line Routing](#) for more information.

You can find or set the side of the source shape that the line leaves using the [SourceDirection property](#) and can find or set the side of the destination shape that the line enters using the [DestinationDirection property](#). The chart above describes each of the direction values.

## Related Topics

### Description

[Drawing Lines](#)

## Connecting Existing Lines to Shapes

Related

The shapes that lines connect to are stored in the [Source](#) and [Destination](#) properties of the `Line_` object. When lines are unconnected, these properties are empty. You can use these properties to change which shapes the line connects or to connect a line that was previously unconnected.

For example, the following statements connect `LineObject` to `ShapeObject1` and `ShapeObject2`.

```
LineObject.Line_.Source = ShapeObject1
LineObject.Line_.Destination = ShapeObject2
```

You can optionally specify the directions that the line enters each shape with the [ReconnectSource](#) and [ReconnectDest](#) methods of the `Line_` object.

**ReconnectSource** (*ShapeObject* [,*ExitDirection*] )  
**ReconnectDest** (*ShapeObject* [,*EnterDirection*] )

These methods connect the line to a shape specified by *ShapeObject*. You can optionally specify the direction the line enters and exits the shapes. The following chart describes each of the direction values.

Value	Direction
0	North
1	East
2	South
3	West

For example, the following statements connect `LineObject` to the bottom of `ShapeObject1` and the top of `ShapeObject2`.

```
LineObject.Line_.ReconnectSource ShapeObject1 0
LineObject.Line_.ReconnectDest ShapeObject2 2
```

## Related Topics

### Description

[Working with Lines](#)

## Setting Line Routing

Related

ABC has several types of routing available for lines.

In ABC you set the routing for new lines to be drawn by clicking one of the line routing buttons with no lines selected. You cannot change the type of routing for lines that have already been drawn.

With ABC OLE Automation, you can specify the type of routing for new lines with the [CurrentLineRouting property](#) of the Chart object. The following table describes the values for the [CurrentLineRouting property](#).

Value	Description
0	Direct
1	Right Angle
2	Curved
3	Org Chart
4	Cause-and-Effect

You can use the [Type property](#) of the Line\_ object to find or set the line routing for a line. The values of the [Type property](#) have the same meaning as [CurrentLineRouting](#), which is described in the table above.

The following statements use Direct line routing to draw a line.

```
Dim NewLine As Object
Dim LineType As Short
Chart.CurrentLineRouting = 0           ' Current line routing is Direct
NewLine = Chart.DrawFreeLine (5, 4)   ' Draw a new line
LineType = NewLine.Line_.Type         ' The new line is Direct
```

## Related Topics

### Description

[Working with Lines](#)

## Formatting Lines

Related

You format lines by changing the color, width, and style of the two ends and the main body (stem) of a line. All line formatting is based on the `Line_` object.

Lines are composed of three parts: the arrow at the start of the line (`SourceArrow`), the main body of the line (`Stem`), and the arrow at the end of the line (`DestArrow`). The properties used to format lines deal with these three parts.

[Line Color](#)

[Line Width](#)

[Line Style](#)

[End Styles](#)

## Related Topics

### Description

[Working with Lines](#)

## Line Color

**Related**

You can color an entire line, including the ends, with one property, or you can color each piece of the line with a separate property. You can use the ABC constants to specify the color. For example, use ABC.Red for the color red. See [Color Representation](#) for more information on specifying colors.

The [Color property](#) colors the entire line, including the ends. The properties [SourceArrowColor](#), [StemColor](#), and [DestArrowColor](#) color individual parts of a line. These properties also can be used to find out the colors of a line.

The following example changes blue lines to red. All other lines are changed to blue stems and red ends.

```
If Line_.Color = ABC.Blue Then
    Line_.Color = ABC.Red
Else
    Line_.SourceArrowColor = ABC.Red
    Line_.DestArrowColor = ABC.Red
    Line_.StemColor = ABC.Blue
End I
```

' If the line stem is blue, ...  
' ...change the entire line to red  
' If the line stem is not blue, ...  
' ...change both ends to red...  
' ...and change the stem to blue

## Related Topics

### Description

[Formatting Lines](#)

## Line Width

Related

You can vary the width of lines and the size of line ends. Varying line widths can distinguish data flows and draw attention to certain transitions or data transfers in a chart.

In ABC you change a selected line's width by clicking the Line tool in the toolbox, clicking an apply button in the ribbon beside the part of the line you want to change (source arrow, stem, or destination arrow), and then choosing a number in the Width box.

With ABC OLE Automation, you use the [SourceArrowSize](#), [StemWidth](#), and [DestArrowSize](#) properties of the Line\_ object to determine the line width. Line width can vary from 1 (hairline) to 5 (thickest).

```
ABCObject.Line_.SourceArrowSize = 2      ' Width of source arrow is 2
ABCObject.Line_.StemWidth = 2           ' Width of stem is 2
ABCObject.Line_.DestArrowSize = 1      ' Width of destination arrow is 1
```

## Related Topics

### Description

[Formatting Lines](#)

## Line Style

Related

You can choose different styles or patterns for lines, including solid and dashed lines.

In ABC you change a selected line's style by clicking the Line tool in the toolbox, and then choosing a style from the Line Style button in the ribbon.

To set or read a line's style with ABC OLE Automation, use the [StemStyle property](#) of the Line\_ object. Set StemStyle to 0 for an invisible line and 1 for a solid line. See [StemStyle](#) for each available style.

The following statements draw a line and then change its style to dotted.

```
Set ABCObject = DrawFreeLine (4, 5)
ABCObject.Line_.StemStyle = 3
```

## Related Topics

### Description

[Formatting Lines](#)

## End Styles

Related

You can choose different styles or patterns for line ends, including arrows, circles, and lines.

In ABC you change a selected line's end style by clicking the Line tool in the toolbox, and then choosing a style from the Line Start or Line End button in the ribbon.

To set or read a line's source arrow style with ABC OLE Automation, use the [SourceArrowStyle property](#) of the Line\_ object. Set [SourceArrowStyle](#) to 0 for no arrow. See [SourceArrowStyle](#) for each available style.

To set or read a line's destination arrow style with ABC OLE Automation, use the [DestArrowStyle property](#) of the Line\_ object. Set [DestArrowStyle](#) to 0 for no arrow. See [DestArrowStyle](#) for each available style.

The following statements draw a line and then change its end styles.

```
Set ABCObject = Chart.DrawFreeLine (4, 5)
ABCObject.Line_.SourceArrowStyle = 2
ABCObject.Line_.DestArrowStyle = 3
```

## Related Topics

### Description

[Formatting Lines](#)

## Displaying Nodes on Connecting Lines

Related

You can display or hide nodes on lines. Nodes appear where lines connect to each other. They help you distinguish between connected lines and lines that merely overlap. Nodes are represented by small, filled circles on lines.

In ABC you display nodes by selecting the Show Nodes on Lines option in the Indicator Options panel of the Preferences dialog box.

With ABC OLE Automation you display or hide nodes using the [ShowNodesOnLines property](#) of the Chart object. Set [ShowNodesOnLines](#) to True to display nodes; False to hide nodes.

```
Chart.ShowNodesOnLines = True      ' Display nodes
Chart.ShowNodesOnLines = False     ' Hide nodes
```

## Related Topics

### Description

[Working with Lines](#)

## Setting Line Crossovers Related

You can set the style and size when lines cross over each other. If you do not choose a crossover style, the lines cross with no indication, which may make it difficult to tell which lines connect which shapes.

In ABC you set the style and size of crossovers by choosing options in the Crossovers area in the Line Options panel of the Preferences dialog box.

You use the [LineCrossoverStyle property](#) of the Chart object to specify the type of crossover. You can specify values for the crossovers as shown in the following table.

Style	Description
0	<span style="border: 1px solid black; padding: 2px;">Related</span> Bunny hops
1	<span style="border: 1px solid black; padding: 2px;">Related</span> Broken lines
2	 Solid lines

You use the [LineCrossoverSize property](#) of the Chart object to specify the size of the crossover when one line crosses another. The setting applies to bunny hops and broken lines, but has no effect when the crossover style is solid lines. The possible values of the property are shown in the following table.

RelativeSize	Description
0	<span style="border: 1px solid black; padding: 2px;">Related</span> Small
1	<span style="border: 1px solid black; padding: 2px;">Related</span> Medium
2	<span style="border: 1px solid black; padding: 2px;">Related</span> Large

For example, the following statements set crossovers to be medium bunny hops.

```
Chart.LineCrossoverStyle = 0      ' Bunny hops
Chart.LineCrossoverSize = 1      ' Medium
```

## Related Topics

### Description

[Working with Lines](#)

## Attaching Text to Lines

Related

Text on lines can describe the flow of information and relationships between connected shapes. You can choose the typeface, size, style, and color of the attached text. When you move a line, the attached text moves with it.

In ABC you attach text to a selected line by clicking the Text tool in the toolbox and then typing.

To attach text to lines using ABC OLE Automation, you create a text block and then attach it to an existing line. See the [Creating Text Blocks](#) and [Drawing Lines](#) for more information on creating text blocks and lines.

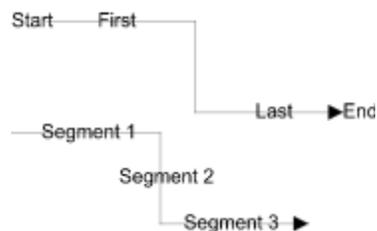
Use the [AttachText method](#) of the Line\_ object to attach text to a line. You can attach the text to the start or end of the line or to any segment of the line.

*LineObject.AttachText TextObject [,SegmentNumber]*

The *LineObject* specifies the line to which you are attaching the text. *TextObject* specifies the text block that you are attaching to the line. The *SegmentNumber* indicates the segment of the line to which the text is to be attached, as defined in the following table.

<b>SegmentNumber</b>	<b>Line Segment</b>
-3	Start
-2	End
-1	First
0	Last
1 through <i>n</i>	The sequential value of the line segment, where <i>n</i> is the number of segments in the line. For example, 1 is the first segment, and 2 is the second segment.

The following illustrations show how the text is placed on a line.



The following example places text on new lines as they are drawn.

```
Sub ABC1_NewLineNOTIFY ()
  Dim TextBlock As Object
  Set TextBlock = ABC1.Chart.DrawTextBlock "Text on a Line"
  ABC1.Object.Line_.AttachText TextBlock -1
End Sub
```

To format text on a line, see [Formatting Text](#).

## Related Topics

### Description

[Working with Lines](#)

## Deleting Lines

Related

There are two ways to delete lines with ABC OLE Automation: delete all lines attached to a specific shape or select a line and clear it or cut it.

Use the [DeleteLines method](#) of the Shape object to delete all the lines attached to a specified shape. Deleting lines with this method does not place the lines in the Windows Clipboard.

For example, the following statement deletes the lines attached to the shape referred to as ABCObject.

```
ABCObject.Shape.DeleteLines
```

See [Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#) for information on clearing and cutting objects.

## Related Topics

### Description

[Working with Lines](#)

Related

## Working with Text

Related

[Creating Text Blocks](#)

[Adding Text to a Shape](#)

[Adding Notes to a Shape](#)

[Attaching Text to a Line](#)

[Formatting Text](#)

[Checking Spelling](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Creating Text Blocks

Related

A text block is a freeform, independent object in a chart. Text blocks are not associated with any shape or line.

In ABC, you create text blocks by first clicking the Text tool in the toolbox. Then you can click anywhere in the chart and type, or you can drag the mouse to create the block and type.

To create a text block using ABC OLE Automation, you first specify the current drawing position, then draw the text block and specify the text string. You access the information about text blocks using the [TextBlock property](#) of the Object object.

You use the [DrawPositionX](#) and [DrawPositionY](#) properties to specify the X and Y coordinates of the upper left corner of the text block. This defines the current drawing position for the chart. The X and Y coordinates are measured from the top left corner of the page. By default, the positions are measured in inches, but you can measure position in centimeters by changing the [Units property](#) of the Preferences object. See [Setting the Current Drawing Position](#) for more information.

Specifying the drawing position is not required, but helps to control the appearance of the chart. Alternatively, you can draw the text block first, and then move the text block to the desired location. See [Moving Objects](#) for more information on moving text blocks.

You use the [DrawTextBlock method](#) with the Chart object to draw a text block at the current position with the specified text string.

Related

Use Chr\$(13) to add a new line to the text string in the [DrawTextBlock method](#). For example, Chart.DrawTextBlock ("Line 1" + Chr\$(13) + "Line 2")

The following example creates a text block two inches from the top of the page and one inch from the left of the page.

```
Dim ABCObject As Object

Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Set ABCObject = Chart.DrawTextBlock ("This is a text block")
```

## Related Topics

### Description

[Working with Text](#)

## Adding Text to a Shape

Related

You can use ABC OLE Automation to add text inside any shape. The text appears inside the text area defined for the shape. Adding text to a shape is equivalent to typing while a shape is selected in ABC.

To add text to a shape, use the [Text property](#) of the Object. The following example creates a shape with text.

```
Dim ABCObject As Object  
  
Set ABCObject = Chart.DrawShape ("External Operation")  
ABCObject.Text = "Text inside a shape"
```

If you are reading the text from a shape, you can use the [TextLF property](#) to preserve the Returns. If you use the [Text property](#), the Returns are changed to spaces.

```
ShapeText = Shape1.TextLF
```

## Related Topics

### Description

[Sizing Shapes to Text](#)

[Working with Text](#)

## Sizing Shapes to Text Related

You can automatically fit shapes to the size of the text inside them. This is especially useful when the length of the text string may be varied and you want to avoid hiding text that will not fit within the shape.

In ABC, you do this by clicking the Text tool and then the Fit Shape button in the ribbon when the shape is selected. Using ABC OLE Automation, you use the [FitShapeToText method](#) for the Shape object.

This example draws a shape, adds text to the shape, and then fits the shape to the text.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape ("Document")
ABCObject.Text = "This is a sample of fitting shapes to text"
ABCObject.Shape.FitShapeToText
```

## Related Topics

### Description

[Adding Text to a Shape](#)

## Adding Notes to a Shape Related

You can attach notes to any shape in a chart. In ABC notes are added in the Note window. With ABC OLE Automation, you do not need to open the Note window to attach notes to a shape.

You use the [NoteText property](#) of the Shape object to attach notes to shapes.

The following example draws a shape, then adds a note to the shape.

```
Dim ShapeObject As Object  
  
Set ShapeObject = Chart.DrawShape ("Document")  
ShapeObject.Text = "Text inside a shape"  
ShapeObject.Shape.NoteText = "This is note text attached to the shape"
```

See the [Formatting Text](#) for information on formatting note text. See [Working with Notes](#) for more information on notes.

If you are reading the note text from a shape, you can use the [NoteTextLF property](#) to preserve the Returns. If you use the [NoteText property](#), the Returns are changed to spaces.

```
NoteText = Shape1.NoteTextLF
```

## Related Topics

### Description

[Working with Text](#)

## Attaching Text to a Line Related

You can attach text to the start or end of a line or to any segment of a line. When text is attached to a line, the text is moved and positioned automatically with the line.

In ABC you can attach text by dragging a text block onto a line or by selecting a line and then clicking the Text tool in the toolbox and typing. To attach text to lines using ABC OLE Automation, you must first create a text block, then attach it to a line.

You use the [DrawTextBlock method](#) to create the text block. See [Creating Text Blocks](#) for information about creating text blocks.

Use the [AttachText method](#) of the Line\_ object to attach a text block to a line. The **AttachText** method has two parameters.

```
LineObject.AttachText TextBlock [,SegmentNumber]
```

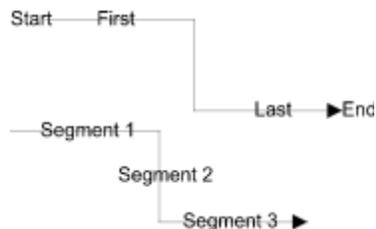
The *TextBlock* parameter is the text block object you are attaching to the line in *LineObject*. The *SegmentNumber* is an optional parameter that indicates which segment of the line will contain the text.

The following table describes each value of *SegmentNumber*.

<b>SegmentNumber</b>	<b>Line Segment</b>
-3	Start
-2	End
-1	First
0	Last
1 through <i>n</i>	The sequential value of the line segment, where <i>n</i> is the number of segments in the line. For example, 1 is the first segment, and 2 is the second segment.

The default value of *SegmentNumber* is -1, which attaches a text block to the center of the first segment of the line.

The following illustrations show the segment on which text is placed based on the *SegmentNumber*.



The following statements draw a line, then create a text block and attach it to the line.

```
Dim LineObject As Object
Dim TextBlock As Object

DrawPositionX = 2
DrawPositionY = 1
Set LineObject = Chart.DrawFreeLine (4, 5)
Set TextBlock = Chart.DrawTextBlock ("Text on a line")
LineObject.Line_.AttachText (TextBlock, 1)
```

### [Unattaching Text from a Line](#)

## Related Topics

### Description

[Working with Text](#)

## Unattaching Text from a Line

Related

You can separate text from a line without deleting the text. This is equivalent to dragging the text away from a line in ABC. Unattaching text from a line has no effect on the line.

To find if text is attached to a line, you use the [AttachedToLine property](#) of the TextBlock object. To unattach text, use the [UnattachFromLine method](#) and move the text to another position. For more information on moving text blocks, see [Moving Objects](#).

The following example draws a line, creates a text block, then attaches the text block to the line. Then the example checks to see if the text is attached to the line, unattaches the text, and moves it to another location.

```
Dim LineObject As Object
Dim TextBlock As Object

DrawPositionX = 2
DrawPositionY = 1
Set LineObject = Chart.DrawFreeLine (4, 5)
Set TextBlock = Chart.DrawTextBlock ("Text on a line")
LineObject.Line_.AttachText (TextBlock, 1) = True

If TextBlock.AttachedToLine Then
    TextBlock.UnattachFromLine
    TextBlock.Left = 3
    TextBlock.Top = 6
End If
```

## Related Topics

### Description

[Attaching Text to a Line](#)

## Formatting Text

You can format a text object in ABC by changing its typeface, its size, its text attributes (such as bold or italic), its color, and its alignment.

All text formatting applies to the entire object. You cannot format a single word differently than the rest of the text in the object.

[Text Typeface and Size](#)

[Bold, Italic, Underline, and Strikethrough](#)

[Text Color](#)

[Text Background](#)

[Text Alignment](#)

## Related Topics

### Description

[Working with Text](#)

## Text Typeface and Size

Related

You can change the typeface and size of text, as well as determine the current typeface and size of the text.

Use the [Size](#) and [Name](#) properties of the Font object to specify the point size and typeface of the text, respectively. You access those properties using the [Font property](#) of the Object object or the [NoteFont property](#) of the Shape object.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Size = 10           ' Text block or text on a line
ABCObject.Font.Size = 10           ' Shape text
ABCObject.Shape.NoteFont.Size = 10 ' Note text
```

The [Size property](#) uses a Long value to specify point sizes, such as 10 or 12.

The [Name property](#) uses a string to determine the typeface. When changing the typeface, ABC matches the highest quality typeface containing the string. For example, `Font.Name = "Roman"` sets the typeface to "Times New Roman" if both "Tms Roman" and "Times New Roman" are available, because "Times New Roman" is a TrueType typeface.

The following example checks to see if a shape's text is Arial 10 pt. If it is, the text changes to Times New Roman.

```
Dim ABCShape As Object

Set ShapeCollection = ABC.ActiveChart.Objects
Set ABCShape = ShapeCollection.ItemFromShapes

Do
    If ABCShape.Font.Name = "Arial" and ABCShape.Font.Size = 10 Then
        ABCShape.Font.Name = "Times New Roman"
    End If
    Set ABCShape = ShapeCollection.ItemFromShapes
Loop While ABCShape.Valid
```

## Related Topics

### Description

[Formatting Text](#)

## Bold, Italic, Underline, and Strikethrough

Related

You can change the attributes of text, such as bold, italic, underline, and strikethrough, as well as determine the current attributes of the text.

The [Bold](#), [Italic](#), [Underline](#), and [Strikethrough](#) properties of the Font object take Boolean values that turn each attribute on or off.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Bold = True           ' Text block or text on a line
ABCObject.Font.Bold = True           ' Shape text
ABCObject.Shape.NoteFont.Bold = True ' Note text
```

The following example creates a text block and makes the text bold and italic.

```
Dim ABCObject As Object

Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Set ABCObject = Chart.DrawTextBlock ("This is a text block")
ABCObject.Font.Bold = True
ABCObject.Font.Italic = True
```

The following example changes all strikethrough text in shapes to underline.

```
Dim ABCShape As Object

Set ShapeCollection = ABC.ActiveChart.Objects
Set ABCShape = ShapeCollection.ItemFromShapes

Do
    If ABCShape.Font.Strikethrough Then
        ABCShape.Font.Strikethrough = False
        ABCShape.Font.Underline = True
    End If
    Set ABCShape = ShapeCollection.ItemFromShapes
Loop While ABCShape.Valid
```

## Related Topics

### Description

[Formatting Text](#)

## Text Color

**Related**

You can change the color of text, as well as determine the current text color. The text color affects only the foreground color of the text; it does not affect the background color. See [Text Background](#) for information on the background of text.

You use the [Color property](#) of the Font object to specify the text color. You can use constants to specify one of the sixteen basic VGA colors, or you can specify the RGB values using the [MakeRGB method](#). See [Using Color](#) for more information on specifying colors.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Color = ABC.BLUE           ' Text block or text on a line
ABCObject.Font.Color = ABC.BLUE           ' Shape text
ABCObject.Shape.NoteFont.Color = ABC.BLUE ' Note text
Chart.FieldFont.Color = ABC.BLUE          ' Field text
Chart.NumberFont.Color = ABC.BLUE         ' Shape numbers
Chart.MasterItems.Date.Font.Color = ABC.BLUE ' Text in the Date Master Item
```

The following example changes all blue text in shapes to red. The color Blue is specified by a constant, and the color Red is specified by its RGB value.

```
Dim ABCShape As Object

Set ShapeCollection = ABC.ActiveChart.Objects
Set ABCShape = ShapeCollection.ItemFromShapes

Do
    If ABCShape.Font.Color = ABC.Blue Then
        ABCShape.Font.Color = ABC.MakeRGB (255, 0, 0)
    End If
    Set ABCShape = ShapeCollection.ItemFromShapes
Loop While ABCShape.Valid
```

## Related Topics

### Description

[Formatting Text](#)

## Text Background

Related

You can make the background behind text opaque or transparent. When a text background is opaque, you cannot see through the text to the objects beneath it. For example, when text is on a line, you can see the line through the text if the background is transparent; you cannot see the line if the background is opaque.

In ABC you make a text background opaque by selecting the text, clicking the Text tool in the toolbox, and selecting the Text Background button in the ribbon. You deselect the Text Background button to make the background transparent.

With ABC OLE Automation, you use the [Opaque property](#) of the Font object to make a text background opaque or transparent.

The Font object contains most formatting properties of text. The way you access the Font object depends on the type of chart text. The following examples show each type of Font object.

```
ABCObject.Font.Opaque = True           ' Text block or text on a line
ABCObject.Font.Opaque = True           ' Shape text
ABCObject.Shape.NoteFont.Opaque = True ' Note text
```

## Related Topics

### Description

[Formatting Text](#)

## Text Alignment

Related

You can align the text inside shapes and in text blocks using the `TextAlignment` property of the object. The [TextAlignment property](#) uses the following Integer values to represent each square in the Alignment Grid in the Text ribbon.

0	1	2
3	4	5
6	7	8

For example, the following line centers the text at the top of the shape.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawShape ("Document")
ABCObject.Text = "This is a sample of aligning text"
ABCObject.TextAlignment = 1
```

The following example creates a text block and left-justifies the text in the text block's vertical center.

```
Dim ABCObject As Object

Set ABCObject = Chart.DrawTextBlock ("This is a sample of aligning text")
ABCObject.TextAlignment = 3
```

## Related Topics

### Description

[Formatting Text](#)

## Checking Spelling



You can check the spelling in ABC by choosing the text you want to check and choosing Spelling in the Edit menu. With ABC OLE Automation, you use the [Spelling method](#) of the Chart object. For example, the following statement starts the Spelling Checker.

```
Chart.Spelling
```

## Related Topics

### Description

[Working with Text](#)

Related

## Working with Data Fields

Related

[Adding Data Fields to a Chart](#)

[Deleting Data Fields from a Chart](#)

[Setting Data Field Preferences](#)

[Working with Data Field Values](#)

[Viewing the Legend](#)

[Using Linked Field Data](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Adding Data Fields to a Chart

Related

In ABC, you can attach a data field table to your chart's shapes. The data field table applies to all shapes in a chart, so any changes you make to the data field descriptions apply to all the shapes. You can store data field tables in templates for use in other charts. You use the [FieldTemplate property](#) to find the FieldTemplate object for a specified FieldValue object.

When you create a link from an existing chart to a new chart, the existing chart's field table is copied to the new chart and the values are accumulated according to the methods you specify. That means you do not have to recreate the data field descriptions in each new chart.

For example, a chart's data fields might be Assigned, Due, and Cost. A specific shape might have the value "Britt Barnes" in the Assigned field, "01/09/95" in the Due field, and "\$1,200" in the Cost field.

1	2	3
Assemble	Finish	Package
Assigned: Britt Barnes	Assigned: Dae Shannon	Assigned: Lana Robertson
Due: 1/9/95	Due: 1/23/95	Due: 1/30/95
Cost: \$1,200.00	Cost: \$2,000.00	Cost: \$800.00

In ABC, you add data fields by choosing the Setup command in the Fields menu to open the Setup Fields dialog box, entering a name for the field, selecting a type for the field, and clicking OK.

In ABC OLE Automation, data fields are stored in the FieldTemplates collection. You use the [FieldTemplates property](#) to find the FieldTemplates collection of the Chart object.

When you create a field using the [Add method](#), it is added to that collection. The [Add method](#) has two parameters. The first is the name of the field. The second parameter, which is optional, sets the type of field. The following table shows the values of the second parameter and their meanings.

Value	Meaning
0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number (default if the parameter is omitted)

You set the name, type, format, accumulation method, and hidden options of a field using properties of the FieldTemplate object as described in [Changing Data Field Attributes](#).

The last statement in the following subroutine adds the object Field1 to the FieldTemplates collection. The field has the field name "Assigned" and is text type because of the 0 value in the second parameter. (If you omit the second parameter, the default is 5, which specifies a number.) The field is created with the default attributes for a text object.

```
Sub Form_Load ()
    Dim ABC As Object
    Set ABC = CreateObject("ABCFlow.application")
    ABC.Visible = True
    Dim Chart As Object
    Set Chart = ABC.New
    Dim Field1 As Object
    Set Field1 = Chart.Fields.Add("Assigned", 0)
End Sub
```

## Changing Data Field Attributes

## Related Topics

### Description

[Working with Data Fields](#)

## Changing Data Field Attributes

**Related**

After you create a field, you can describe its attributes using the [Name](#), [Type](#), [Format](#), [AccumulationMethod](#), and [Hidden](#) properties of the FieldTemplate object. These properties correspond to the options in the Setup Fields dialog box.

The name of a field appears in the chart next to the field value. The [Name property](#) lets you rename a field, such as "Task 1" You name a field at the time you create it using the [Add method](#), so you only use this property to change the name of a field.

Data fields can have one of six major types.

Value	Type
0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number (default if the parameter is omitted)

Most of the major types have choices within them. For example, dates can be M/D/YY (8/1/94), MMM-D-YY (Aug-1-94), MMMM DD, YYYY (August 01, 1994), and so forth. The [Format property](#) lets you specify the format of the field.

Value	Duration Format	Value	Date Format
100	# w.	200	M/D/YY
101	# weeks	201	MMMM-D-YY
102	# d.	202	MMMM DD, YYYY
103	# days	203	MMM-YY
104	# h.	204	MMMM YYYY
105	# hrs.		
106	# hours	Value	Currency Format
107	# m.	300	###0.00(###0.00)
108	# mins.	301	#,##0.00(#,##0.00)
109	# minutes	302	###0(###0)
110	# s.	303	#,##0(#,##0)
111	# secs.		
112	# seconds	Value	Number Format
113	# TMU	500	###0
114	h:m	501	###0.00
115	m:s	502	###0.0000
116	h:m:s	503	#,##0
		504	#,##0.00
		505	#,##0.0000
Value	Percent Format		
400	##%		
401	#0.00%		

The [AccumulationMethod property](#) lets you specify the type of accumulation used to calculate the Legend values and the Linked Fields values in any linked shapes.

Value	Accumulation Method
0	No Accumulation
1	Sum
2	Mean
3	Median

- 4 Min
- 5 Max
- 6 Range
- 7 Total
- 8 Non-Null Total

The [Hidden property](#) lets you specify whether a field and its value are displayed in the chart. Set the [Hidden property](#) to True to hide the field or to False to display the field.

For example, the following subroutine, which assumes that there is a field named Cost, changes the field's attributes so that it is named Expense, changes the type and format to text, sets the accumulation method to No Accumulation, and makes it hidden.

```
Sub ABC1_ChartChangeNOTIFY ()
    Dim Chart As Object
    Set Chart = ABC.ActiveChart

    Dim CurrentField As Object
    Set CurrentField = Chart.FieldTemplates.Item("Cost")
    CurrentField.Name = "Expense"
    CurrentField.Type = 0
    CurrentField.Format = 0
    CurrentField.AccumulationMethod = 0
    CurrentField.Hidden = True
End Sub
```

## Related Topics

### Description

[Adding Data Fields to a Chart](#)

## Deleting Data Fields from a Chart Related

Deleting a data field removes it from every shape in the chart and deletes its values. You delete data fields using the [DeleteField method](#) in the FieldTemplates collection.

*FieldTemplatesCollection.DeleteField* *FieldTemplateObject*

With the [DeleteField method](#), you provide the FieldTemplate object as a parameter. For example, the following statement deletes the field object contained in CurrentField.

```
Chart.FieldTemplates.DeleteField CurrentField
```

The following subroutine, which assumes that there are data fields, uses the [Count property](#), the [Item method](#), and the [DeleteField method](#). It searches through the data fields in the FieldTemplates collection and deletes any that have a type of 3 (currency).

```
Sub ABC1_ChartCloseSUBCLASS ()
    Dim Chart As Object
    Dim FieldTemplates As Object
    Set Chart = ABC.ActiveChart

    Dim CurrentField As Object
    Set FieldTemplates = Chart.FieldTemplates
    For FieldNumber = 1 to FieldTemplates.Count
        Set CurrentField = FieldTemplates.Item(FieldNumber)
        If CurrentField.Type = 3 Then
            FieldTemplates.DeleteField CurrentField
        End If
    Next FieldNumber
End Sub
```

See the [Finding Objects in a Chart](#) for more information on using collections.

## Related Topics

### Description

[Working with Data Fields](#)

## Setting Data Field Preferences

Related

You can set preferences for data fields, such as the font used to display data fields in a chart and the placement of data fields relative to shapes.

In ABC, you set data field preferences by choosing Preferences in the Fields menu and then choosing options in the Preferences dialog box. All the data field preferences in this dialog box are also available using ABC OLE Automation.

The [FieldNamesHidden property](#) of the Chart object lets you choose whether you want to show the data field names in a chart. Values in the data field are not affected by this property. Set the [FieldNamesHidden property](#) to True to hide the field names or to False to display them.

The [FieldPlacement property](#) of the Chart object lets you prescribe where to position data fields in relation to their associated shapes. You use the values in this table.

Value	Field Placement
0	Left
1	Right
2	Above
3	Below
4	Inside Top
5	Inside Middle

The [FieldsOpaque property](#) of the Chart object lets you choose whether the background of data fields is opaque.

The [FieldsHoursPerDay property](#) of the Chart object lets you set the number of hours in a workday. This value is used when a field is converted between hours and days. For example, the value is used if you change the data field's format from hours to days or you link to a chart that displays data fields in a different format.

The [FieldsDaysPerWeek property](#) of the Chart object lets you set the number of days in a workweek. This value is used when a data field is converted between days and weeks when totaling durations.

The [FieldFont property](#) of the Chart object returns the Font object used for data fields. You can format data field text just as you format other text objects. The formatting for fields applies to all fields in the chart and to the text in the Legend.

The [FieldFont property](#) contains the following properties.

Property	Font Description
<a href="#">Bold</a>	True if text is <b>bold</b> ; False if text is not bold
<a href="#">Italic</a>	True if text is <i>italic</i> ; False if text is not italic
<a href="#">Strikethrough</a>	True if text is <del>strikethrough</del> ; False if text is not strikethrough
<a href="#">Underline</a>	True if text is <u>underline</u> ; False if text is not underline
<a href="#">Opaque</a>	True if text is opaque; False if text is not opaque
<a href="#">Name</a>	The typeface name of the font
<a href="#">Size</a>	The point size of the font

See [Formatting Text](#) for more information on these properties.

The following example sets preferences for data fields in a chart.

```
Dim ABC As Object
Dim Chart As Object
```

```
Set ABC = CreateObject ("ABCFlow.application")
Set Chart = ABC.ActiveChart
```

```
Chart.FieldFont.Name = "Arial"
Chart.FieldFont.Size = 12
Chart.FieldFont.Bold = True
Chart.FieldFont.Italic = True
Chart.FieldFont.Strikethrough = False
Chart.FieldFont.Underline = False
Chart.FieldFont.Opaque = False
```

## Related Topics

### Description

[Working with Data Fields](#)

## Working with Data Field Values

Related

You can enter values into data fields for any shape in a chart. You do not have to enter values for all data fields or all shapes.

In ABC, you enter values in data fields by clicking the Field Viewer button and working in the Field Viewer dialog box.

The FieldValues collection of the Object object contains the data field values. To enter values, you specify the values using the [Value property](#) of the FieldValue object and the [Item method](#) of the FieldTemplates collection. You access the values of data fields using the [FieldValues property](#) of the Object object.

For example, the following statements set values for an existing shape, Shape1. The statements assume that the chart has the fields Assigned, Due, and Cost.

```
Shape1.FieldValues.Item("Assigned").Value = "Beginning"  
Shape1.FieldValues.Item("Due").Value = "1/3/95"  
Shape1.FieldValues.Item("Cost").Value = "200"
```

You also can read values. For example, the following subroutine reads the Profit in all shapes and turns the text red for all that are negative.

```
Sub ABC1_DoubleClickSUBCLASS ()  
    Dim ABC As Object  
    Dim Chart As Object  
    Set Chart = ABC.ActiveChart  
  
    Dim AllShapes As Object  
    Set AllShapes = Chart.Objects  
  
    Dim Shape As Object  
    For ChartCount = 1 to AllShapes.Count  
        Set Shape = AllShapes.ItemFromShapes(ChartCount)  
        If Shape.FieldValues.Item("Profit").Value < 0 Then  
            Shape.Font.Color = ABC.RED  
        End If  
    Next ChartCount  
End Sub
```

You can read or set the day, month, and year of a Date field (Type = 2) using the [Day property](#), [Month property](#), and [Year property](#) of the FieldValue object. For example, the following statements change any October 15 due dates to October 17.

```
Set Field = Shape1.FieldValues.Item("Due")  
If Field.Month = 10 And Field.Day = 15 Then Field.Day = 17
```

You can find out if a data field contains any value using the [IsEmpty property](#) of the FieldValue object. For example, the following statement puts into the variable NoCostExists whether the existing "Cost" data field for Shape1 is empty.

```
NoCostExists = Shape1.FieldValues.Item("Cost").IsEmpty
```

You can read the format of the value in a data field using the [FormattedValue property](#) of the FieldValue object. For example, the following statement puts the formatted value from the existing "Cost" data field for Shape 1 into the variable CostFormat.

```
CostFormat = Shape1.FieldValues.Item("Cost").FormattedValue
```

You use the [Accumulation property](#) in the FieldTemplate object to return the accumulated value for a specific field. For example, the following statements calculate the accumulated value of the field template

created as Cost in TotalCost.

```
Dim TotalCost As Double  
TotalCost = Cost.Accumulation
```

The [Accumulation property](#) is read only.

[Knowing When Data Fields Change](#)  
[Opening the Field Viewer](#)

## Related Topics

### Description

[Working with Data Fields](#)

## Knowing When Data Fields Change

Related

Sometimes you want to know when the user has changed a data field value so you can react to that change. You can use the [FieldValueChangedNOTIFY event](#) to be notified when a field has changed.

**Note:** For ABC1 VBX notification to function, you must register events using the [RegisterEvent method](#) of the Application object. (For more information on registering events, see [Handling ABC Events](#).) For example, the following statement registers the [FieldValueChangedNOTIFY event](#).

```
ABC.RegisterEvent ABC1.VBX, Form1.Caption, "FieldValueChangedNOTIFY"
```

When you are notified with the [FieldValueChangedNOTIFY event](#) that a field has changed, you can use the FieldValue object of the VBX (ABC1.FieldValue) to access that field. ABC1.Object tells you which shape or line had the change made in the field value. (ABC1.Chart tells you which chart the object is in.)

The following subroutine checks any changed field. If the field is the cost and if it is zero, then the object changes to red.

```
Sub ABC1_FieldValueChangedNOTIFY ()
  If ABC1.FieldValue.Name = "Cost" And ABC1.FieldValue.Value = 0 Then
    ABC1.Object.Color = ABC1.App.RED
  End If
```

## Related Topics

### Description

[Working with Data Field Values](#)

## Opening the Field Viewer Related

The Field Viewer dialog box is used in ABC to enter and display data in fields for a selected shape.

You open the Field Viewer using the [FieldViewerVisible property](#) in the Application object. Set the property to True to display the Field Viewer or to False to hide it.

```
ABC.FieldViewerVisible = True      ' Field Viewer is open
ABC.FieldViewerVisible = False     ' Field Viewer is closed
```

## Related Topics

### Description

[Working with Data Field Values](#)

## Viewing the Legend Related

The Legend in ABC shows the totals of the data fields in a chart. The totals reflect the current state of the chart and update automatically when any field changes.

The totals in the Legend have the same font and style as other data field fonts. See [Setting Data Field Preferences](#) for information about formatting field data.

Fields with the accumulation method No Accumulation do not appear in the Legend. See [Changing Data Field Attributes](#) for information on setting a data field's accumulation method.

In ABC, you show or hide the Legend by opening the Fields menu and choosing Show Legend or Hide Legend. In ABC OLE Automation, you use the [ShowLegend property](#) of the Chart object to show or hide the Legend or to determine whether the Legend is already displayed. When [ShowLegend](#) is True, the Legend displays; when it is False, the Legend is hidden.

The following statement shows the Legend.

```
Chart.ShowLegend = True
```

Use the following statement to hide the Legend.

```
Chart.ShowLegend = False
```

## Related Topics

### Description

[Working with Data Fields](#)

## Using Linked Field Data Related

One of the most useful features of ABC is linked charts. Linking charts lets you have a top-level chart showing only summaries, and then go quickly to the linked charts to see details that would otherwise obscure the overall picture. You can find information about linking using the help system for ABC FlowCharter.

The [LinkFields property](#) in the Shape object returns True if the object's field data show the accumulation of the field data in the linked chart. For example, the following statements set the value of `LinkedData` to the value of the `Cost` field if the object is linked to another chart and shows the information from that chart.

```
If Shape1.LinkFields Then
    LinkedData = Shape1.Fields.Item("Cost").Value
End If
```

The [UpdateFields method](#) in the Chart object updates all the fields for all the linked shapes in a chart so they reflect the values in the linked charts. It is the equivalent of opening the Fields menu in ABC and choosing the Update command. For example, the following statement updates the fields for all shapes in the chart that are linked to another chart.

```
Chart.UpdateFields
```

For information on the [LinkNOTIFY event](#), the [LinkIndicator property](#), the [LinkShadow property](#), the [IsLaunched property](#), the [IsLinked property](#), the [LinkedChartName property](#), and the [Link method](#), see [Linking Charts](#).

You can empty data fields using the [Empty method](#) of the `FieldValue` object. For example, the following statement empties the value from the existing "Assigned" data field for `Shape1`.

```
Shape1.FieldValues.Item("Assigned").Empty
```

## Related Topics

### Description

[Working with Data Fields](#)

Related

## Using Color

Related

[Color representation](#)

[Setting shape colors](#)

[Setting line colors](#)

[Setting text colors](#)

## Related Topics

### Description

[Contents](#)

[Properties and Methods by Task](#)

## Color Representation

Related

You can use ABC OLE Automation to set the colors of ABC objects and text. You can set colors using several different formats. The format you choose depends on what you are trying to accomplish.

[Color Constants Description](#)

[BasicColor Method Description](#)

[RGB Values](#)

[Color Double Values](#)

[Color Equivalents](#)

## Related Topics

### Description

[Working with Color](#)

## Color Constants Description

Related

When you want to set colors quickly and want to choose only among the sixteen VGA colors, you can use the colors whose names are defined as constants in ABC OLE Automation, such as ABC.BLUE. The following table lists the color constants.

Color	Name
White	WHITE
Black	BLACK
Red	RED
Green	GREEN
Blue	BLUE
Yellow	YELLOW
Magenta	MAGENTA
Cyan	CYAN
Gray	GRAY
Dark Red	DK_RED
Dark Green	DK_GREEN
Dark Blue	DK_BLUE
Dark Yellow	DK_YELLOW
Dark Magenta	DK_MAGENTA
Dark Cyan	DK_CYAN
Dark Gray	DK_GRAY

For example, suppose you want to set the color of shape numbers to red. The following statements do that.

```
Dim ABC As Object
Set ABC = CreateObject (ABCFlow.Application)
ABC.ActiveChart.NumberFont.Color = ABC.RED
```

It does not matter what variable you use for the ABC application. The "ABC" in the previous statements depends on the **Dim**. The following statements have exactly the same effect.

```
Dim ABCApplication As Object
Set ABCApplication = CreateObject (ABCFlow.Application)
ABCApplication.ActiveChart.NumberFont.Color = ABCApplication.RED
```

**Note:** You cannot change the values of the constants. For example, you cannot make the constant **DK\_YELLOW** yield the color red.

## Related Topics

### Description

[Color Constants Description](#)

[Color Equivalents](#)

[Color Representation](#)

## BasicColor Method Description

Related

A quick way to set the color is using the array of colors in the [BasicColor method](#) of the Application object. As with the defined constants, the [BasicColor method](#) lets you set colors from the sixteen VGA colors. For example, the following statement sets the color of shape numbers color to blue.

```
ABC.ActiveChart.NumberFont.Color = ABC.BasicColor(4)
```

The following chart lists the [BasicColor method](#) values.

Color	BasicColor
White	0
Black	1
Red	2
Green	3
Blue	4
Yellow	5
Magenta	6
Cyan	7
Gray	8
Dark Red	9
Dark Green	10
Dark Blue	11
Dark Yellow	12
Dark Magenta	13
Dark Cyan	14
Dark Gray	15

**Note:** You cannot change the values in the [BasicColor method](#). For example, you cannot make **BasicColor**(10) yield the color purple.

## Related Topics

### Description

[Color Constants Description](#)

[Color Equivalents](#)

[Color Representation](#)

## RGB Values

Related

If you want to use a color that is not one of the 16 defined in ABC OLE Automation, you can specify the color as quantities of red, green, and blue. You specify each color with a number from 0 (no color) through 255 (solid color). By specifying values for red, green, and blue, you can choose from over 16 million colors. You specify the colors using the [MakeRGB method](#) of the Application object.

For example, **MakeRGB(0,0,255)** is no red, no green, and solid blue, so it specifies blue.

**MakeRGB(255,255,0)** is solid red, solid green, and no blue, so it specifies yellow. **MakeRGB(127,0,255)** is 50% red, no green, and solid blue, so it specifies a purple color. For example, the following statement sets the color of shape numbers to purple.

```
ABC.ActiveChart.NumberFont.Color = ABC.MakeRGB(127,0,255)
```

## Related Topics

### Description

[Color Constants Description](#)

[Color Equivalents](#)

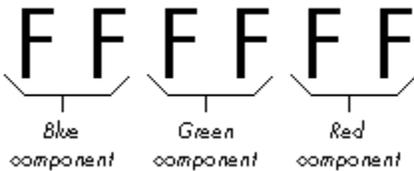
[Color Representation](#)

## Color Double Values

**Related**

You can specify a color as a double. You probably only want to use this method when you are passed a color from another application.

In this method, the color equals the decimal equivalent of six hexadecimal digits. The first two hexadecimal digits set the blue component, the second two set the green component, and the third two set the red component. (Notice that the order is reversed from when you are using [MakeRGB](#).)



For example, FF0000 in hexadecimal means solid blue, no green, and no red. If you set an object to &HFF0000 (or to 16777216, the decimal equivalent of &HFF0000), the object is set to blue.

Most programmers work in hexadecimal rather than decimal for colors, using the Visual Basic language element &H to specify that the following number is in hexadecimal, but you can use whichever makes you most comfortable. For example, the following statements both set the color of shape numbers to blue.

```
ABC.ActiveChart.NumberFont.Color = &HFF0000  
ABC.ActiveChart.NumberFont.Color = 16777216
```

**Related**

You can use the Windows Calculator to convert between hexadecimal and decimal numbers. After choosing Scientific in the Calculator's View menu, type a number and select Hex or Dec to convert it. See your Windows documentation for more information.

## Related Topics

### Description

[Color Constants Description](#)

[Color Equivalents](#)

[Color Representation](#)

[RGB Values](#)

## Color Equivalents

**Related**

The following table shows the equivalents for the sixteen VGA colors for the four color methods. You can set over sixteen million different colors using [MakeRGB](#) or double values.

Color	Name	BasicColor	MakeRGB	Double (Decimal)	Double (Hex)
White	WHITE	0	(255,255,255)	16777215	FFFFFF
Black	BLACK	1	(0,0,0)	0	0
Red	RED	2	(255,0,0)	255	FF
Green	GREEN	3	(0,255,0)	65280	FF00
Blue	BLUE	4	(0,0,255)	16711680	FF0000
Yellow	YELLOW	5	(255,255,0)	65535	FFFF
Magenta	MAGENTA	6	(255,0,255)	16711935	FF00FF
Cyan	CYAN	7	(0,255,255)	16776960	FFFF00
Gray	GRAY	8	(192,192,192)	12632256	C0C0C0
Dark Red	DK_RED	9	(127,0,0)	127	7F
Dark Green	DK_GREEN	10	(0,127,0)	32512	7F00
Dark Blue	DK_BLUE	11	(0,0,127)	8323072	7F0000
Dark Yellow	DK_YELLOW	12	(127,127,0)	326397	7F7F
Dark Magenta	DK_MAGENTA	13	(127,0,127)	8323199	7F007F
Dark Cyan	DK_CYAN	14	(0,127,127)	8355584	7F7F00
Dark Gray	DK_GRAY	15	(127,127,127)	8355711	7F7F7F

The following five statements all do the same thing. Each changes the shape's fill color to blue.

```
Object.Shape.FillColor = ABC.BLUE
Object.Shape.FillColor = ABC.BasicColor(4)
Object.Shape.FillColor = ABC.MakeRGB(0,0,255)
Object.Shape.FillColor = 16711680
Object.Shape.FillColor = &HFF0000
```

You can use the [MakeRGB method](#) of the Application object to find the double equivalent of RGB values. For example, the following statement puts the double value for blue (16711680) into the variable CurrentColor.

```
Dim CurrentColor As Long
CurrentColor = ABC.MakeRGB(0,0,255)
```

## Related Topics

### Description

[Color Constants Description](#)

[BasicColor Method Description](#)

[RGB Values](#)

[Color Double Values](#)

[Color Representation](#)

## Setting Shape Colors

Related

You can color a shape by setting its fill color, its border color, and its shadow color.

You set the fill color for shapes using the [FillColor property](#) of the Shape object or the [Color property](#) of the Object object. Both properties produce the same effect. For example, the following statements draw a shape and then change its fill color to blue using the [FillColor property](#). They then change its fill color to red using the [Color property](#).

```
Dim NewObj1 As Object
Set NewObj1 = Chart.DrawShape
NewObj1.Shape.FillColor = ABC.BLUE
NewObj1.Color = ABC.RED
```

You set the border color for shapes using the [BorderColor property](#) of the Shape object. For example, the following statement makes the border of a shape green.

```
NewObj1.Shape.BorderColor = ABC.GREEN
```

You set the shadow color for shapes using the [ShadowColor property](#) of the Shape object. For example, the following statement makes the shadow of a shape gray.

```
NewObj1.Shape.ShadowColor = ABC.GRAY
```

## Related Topics

### Description

[Working with Color](#)

## Setting Line Colors

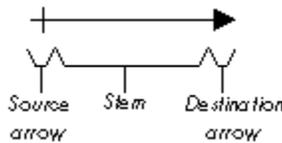
**Related**

You set the colors for lines by coloring the entire line at once or by coloring parts of the line. You use the [Color property](#) to set the color for the entire line, including the source arrow, the stem, and the destination arrow.

You can use the [Color property](#) with either the Line\_ object or the Object object. Using either object produces the same effect. For example, the following statements draw a line and then change its color to blue using the Line\_ object. They then change the line's color to red using the Object object.

```
Dim NewObj1 As Object
Set NewObj1 = DrawFreeLine (3,4)
NewObj1.Line_.Color = ABC.BLUE
NewObj1.Color = ABC.RED
```

You can color parts of a line by coloring the source arrow, the destination arrow, and the stem.



You color the arrow at the source of a line using the [SourceArrowColor property](#) of the Line\_ object. For example, the following statement makes a source arrow red.

```
NewObj1.Line_.SourceArrowColor = ABC.RED
```

You color for stem of a line (the part excluding the destination arrow and source arrow) using the [StemColor property](#) of the Line\_ object. For example, the following statement makes a stem yellow.

```
NewObj1.Line_.StemColor = ABC.YELLOW
```

You color the arrow at the destination of a line using the [DestArrowColor property](#) of the Line\_ object. For example, the following statement makes a destination arrow blue.

```
NewObj1.Line_.DestArrowColor = ABC.BLUE
```

## Related Topics

### Description

[Working with Color](#)

## Setting Text Colors

**Related**

You can set the color of all the text objects that can occur in ABC. You set the color for text using the [Color property](#) of the Font object.

There are variations in how you set text color depending on where the text occurs. The following examples show each type of text object.

```
Object.Font.Color = ABC.BLUE           ' Text block or text on a line
Object.Font.Color = ABC.BLUE           ' Shape text
Object.Shape.NoteFont.Color = ABC.BLUE ' Note text
Chart.FieldFont.Color = ABC.BLUE       ' Field text
Chart.NumberFont.Color = ABC.BLUE      ' Shape numbers
Chart.MasterItems.Date.Font.Color = ABC.BLUE ' Text in the Date master item
```

You set the color of all the MasterItems text objects in the same way as the [Date](#) object in the above example. To set the text color for the [ChartName](#), [Logo](#), [PageNumber](#), [Text1](#), [Text2](#), and [Time](#) MasterItems objects, replace [Date](#) in the above example with the appropriate MasterItems object.

For more information on formatting text, see [Formatting Text](#).

## Related Topics

### Description

[Working with Color](#)

**Related**

## Language Reference

**Related**

[Properties, alphabetical](#)

[Methods, alphabetical](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Events, alphabetical](#)

[All Properties, Methods, Objects, and Events, alphabetical](#)

[Properties and Methods by Task](#)

[ABC Menu Command equivalents](#)

### Reference Description

In the descriptions of the ABC OLE Automation procedures, methods, and events, each element has the following entries, as appropriate.

**Related**

Steps in the ABC application that are equivalent to using the property, method, or event. Click the ABC Equivalent button at the right of the topic title to jump to the ABC application help topic that describes the equivalent.

**Example**

A concise programming use of the element. Click the button at the right of the topic title and the example appears in a separate window that is always on top of all other windows. Click the Close button to close the window. Click the Copy button to put the example in the Windows Clipboard so you can paste it into your program. **Note:** You should [maximize](#) the example window before you copy to the Clipboard to avoid unexpected wraps in longer statements. Some long lines may still wrap, so be sure to unwrap them in your program.

**Related**

Click the button at the right of the topic title to see elements that are related to the element, topics that describe the element, and objects that contain the element. You can click the jump terms to go to the related topics.

**Usage** The syntax for using the element in a program. You must replace items in *italic* with the appropriate variable from a Dim statement.

**Description** How you might use the element.

**Data Type** The type of the element (Object, Collection, Event, String, Integer, Double, Long, Variant).

**Value** The values that you can set the element to or that the element returns.

## Related Topics

Description

[Contents](#)

[Conventions](#)

## Properties

Related

A

B

C

Related

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

### A

[Accumulation](#)

[AccumulationMethod](#)

[ActiveChart](#)

[AlignToRulers](#)

[Application](#)

[AttachedToLine](#)

### B

[Bold](#)

[BorderColor](#)

[BorderStyle](#)

[BorderWidth](#)

[Bottom \(Object object\)](#)

[Bottom \(Application object\)](#)

### C

[Caption](#)

[CenterX](#)

[CenterY](#)

[ChannelAlignment](#)  
[ChartName](#)  
[ChartNameShown](#)  
[Charts](#)  
[Checked](#)  
[ClipboardFormatAvailable](#)  
[Color \(Object object\)](#)  
[Color \(Font object\)](#)  
[Color \(Line\\_ object\)](#)  
[Count](#)  
[CurrentLineRouting](#)  
[CurrentShape](#)  
[CurrentShapePalette](#)

## **D**

[Date](#)  
[DateShown](#)  
[DateStyle](#)  
[DefaultFilePath](#)  
[DestArrowColor](#)  
[DestArrowSize](#)  
[DestArrowStyle](#)  
[Destination](#)  
[DestinationDirection](#)  
[DrawDirection](#)  
[DrawPositionX](#)  
[DrawPositionY](#)  
[DrawSpacingX](#)  
[DrawSpacingY](#)

## **E**

[Enabled](#)

## **F**

[FieldFont](#)  
[FieldNamesHidden](#)  
[FieldPlacement](#)  
[FieldsDaysPerWeek](#)  
[FieldsHoursPerDay](#)  
[FieldsOpaque](#)  
[FieldTemplate](#)  
[FieldTemplates](#)  
[FieldValues](#)  
[FieldViewerVisible](#)  
[FieldViewerWindowHandle](#)  
[FillColor](#)  
[FillPattern](#)  
[Font](#)  
[Format](#)  
[FormattedValue](#)  
[FullName \(Application object\)](#)  
[FullName \(Chart object\)](#)

## **G**

[GuidelinesOn](#)

## **H**

[HasDiskFile](#)

[Height \(Object object\)](#)  
[Height \(Application object\)](#)  
[Height \(PageLayout object\)](#)  
[Hidden](#)  
[Hourglass](#)

## I

[IndexVisible](#)  
[IndexWindowHandle](#)  
[IsEmpty](#)  
[IsLaunched](#)  
[IsLinked](#)  
[Italic](#)

## L

[LaunchCommand](#)  
[LaunchFlags](#)  
[LaunchIndicator](#)  
[LaunchShadow](#)  
[LaunchStartDir](#)  
[Left \(Object object\)](#)  
[Left \(Application object\)](#)  
[Line](#)  
[LineCrossoverSize](#)  
[LineCrossoverStyle](#)  
[LineSpacingX](#)  
[LineSpacingY](#)  
[LinkedChartName](#)  
[LinkFields](#)  
[LinkIndicator](#)  
[LinkShadow](#)  
[Logo](#)  
[LogoShown](#)  
[LogoPathname](#)

## M

[MarginBottom](#)  
[MarginLeft](#)  
[MarginRight](#)  
[MarginTop](#)  
[MasterItems](#)

## N

[Name \(Application object\)](#)  
[Name \(Chart object\)](#)  
[Name \(FieldTemplate object\)](#)  
[Name \(FieldValue object\)](#)  
[Name \(Font object\)](#)  
[NextNumber](#)  
[NextShapeHeight](#)  
[NextShapeWidth](#)  
[NoRepaint](#)  
[NoteFont](#)  
[NoteIndicator](#)  
[NoteShadow](#)  
[NoteText](#)  
[NoteTextLF](#)

[NoteViewerVisible](#)  
[NoteViewerWindowHandle](#)  
[Number](#)  
[NumberFont](#)  
[NumberShown](#)

## **O**

[Objects](#)  
[ObjectType](#)  
[OLE](#)  
[Opaque](#)  
[OperatingSystem](#)  
[Orientation](#)

## **P**

[PageCount](#)  
[PageHeight](#)  
[PageLayout](#)  
[PageNumber](#)  
[PageNumberShown](#)  
[PageOrder](#)  
[PageWidth](#)  
[PaperSize](#)  
[Parent](#)  
[Path](#)  
[PercentGaugeValue](#)  
[Preferences](#)  
[PrintBlankPages](#)  
[Printer](#)  
[Protected](#)

## **R**

[Range](#)  
[ReadOnly](#)  
[Right \(Object object\)](#)  
[Right \(Application object\)](#)

## **S**

[Saved](#)  
[ScrollLeft](#)  
[ScrollTop](#)  
[Selected](#)  
[SelectedLineCount](#)  
[SelectedObjectCount](#)  
[SelectedOtherCount](#)  
[SelectedShapeCount](#)  
[ShadowColor](#)  
[ShadowOffset](#)  
[ShadowStyle](#)  
[Shape](#)  
[ShapeName](#)  
[ShapePaletteVisible](#)  
[ShapePaletteWindowHandle](#)  
[ShapeSizing](#)  
[ShowLegend](#)  
[ShowNodesOnLines](#)  
[ShowRulers](#)

[Size](#)  
[SmartShapeSpacing](#)  
[Source](#)  
[SourceArrowColor](#)  
[SourceArrowSize](#)  
[SourceArrowStyle](#)  
[SourceDirection](#)  
[SSSHorizontal](#)  
[SSSVERTICAL](#)  
[StatusBar](#)  
[StemColor](#)  
[StemStyle](#)  
[StemWidth](#)  
[StretchType](#)  
[Strikethrough](#)

## **T**

[Text \(Object object\)](#)  
[Text \(Menu collection\)](#)  
[Text \(MenuItem object\)](#)  
[Text1](#)  
[Text1Shown](#)  
[Text2](#)  
[Text2Shown](#)  
[TextAlignment](#)  
[TextBlock](#)  
[TextLF](#)  
[Time](#)  
[TimeShown](#)  
[Top \(Object object\)](#)  
[Top \(Application object\)](#)  
[TouchAlignment](#)  
[Type \(Chart object\)](#)  
[Type \(FieldTemplate object\)](#)  
[Type \(FieldValue object\)](#)  
[Type \(Line object\)](#)  
[Type \(Object object\)](#)  
[TypeRequiresEXE](#)  
[TypeUsesEXE](#)

## **U**

[Underline](#)  
[UndoAvailable](#)  
[UniqueID](#)  
[Units \(Chart object\)](#)  
[Units \(Preferences object\)](#)

## **V**

[Valid](#)  
[Value](#)  
[Version](#)  
[View](#)  
[Visible \(Application object\)](#)  
[Visible \(Menu collection\)](#)

## **W**

[Width \(Object object\)](#)

Width (Application object)  
Width (PageLayout object)  
WindowHandle

**Z**

ZoomPercentage

## Related Topics

Description

[Language Reference](#)

## Methods

Related

A

B

C

Related

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

### A

[Activate \(Application object\)](#)

[Activate \(Chart object\)](#)

[Add \(Charts collection\)](#)

[Add \(FieldTemplates collection\)](#)

[AddFromTemplate](#)

[AddHorizontalGuideline](#)

[AddMenu](#)

[AddVerticalGuideline](#)

[AppendItem](#)

[ArrangeIcons](#)

### B

[BasicColor](#)

### C

[CancelFullScreen](#)

[CascadeCharts](#)

[ChartTypeShutdown](#)

[Clear](#)

[ClearGuidelines](#)

[CloseAll](#)

[CloseChart](#)

[Copy](#)

[CreateAddOn](#)

[Cut](#)

## **D**

[DeleteAll](#)

[DeleteField](#)

[DeleteItem](#)

[DeleteLines](#)

[DeselectAll](#)

[DoVerb](#)

[DrawFreeLine](#)

[DrawLine](#)

[DrawLineToOneObject](#)

[DrawShape](#)

[DrawTextBlock](#)

[Duplicate \(Object object\)](#)

[Duplicate \(Chart object\)](#)

## **E**

[Empty](#)

## **F**

[FitShapeToText](#)

[FullScreen](#)

## **G**

[GroupAndLink](#)

## **H**

[Help](#)

[HideAll](#)

[HidePercentGauge](#)

[Hint](#)

## **I**

[InsertItem](#)

[InsertObjectFromFile](#)

[Item \(Objects collection\)](#)

[Item \(Charts collection\)](#)

[Item \(FieldTemplates collection\)](#)

[Item \(FieldValues collection\)](#)

[Item \(Menu collection\)](#)

[ItemFromAll](#)

[ItemFromAttachments](#)

[ItemFromFieldValue](#)

[ItemFromLines](#)

[ItemFromNumber](#)

[ItemFromSelection](#)

[ItemFromShapes](#)

[ItemFromText](#)

[ItemFromUniqueID](#)

## **L**

[Launch](#)

[Link](#)

## **M**

[MakeRGB](#)  
[Maximize \(Application object\)](#)  
[Maximize \(Chart object\)](#)  
[Minimize \(Application object\)](#)  
[Minimize \(Chart object\)](#)  
[MsgBox](#)

## **N**

[New](#)  
[NewFromTemplate](#)

## **O**

[Open](#)

## **P**

[Paste](#)  
[PasteLink](#)  
[PasteSpecial](#)  
[PercentGauge](#)  
[PercentGaugeCancelled](#)  
[PrintOut](#)  
[PrintSelected](#)

## **Q**

[Quit](#)

## **R**

[RegisterEvent](#)  
[RemoveAddOn](#)  
[RemoveMenu](#)  
[ReNUMBER](#)  
[Repaint](#)  
[ReplaceShape](#)  
[Restore \(Application object\)](#)  
[Restore \(Chart object\)](#)  
[RestorePicture \(Object Object\)](#)  
[RestorePicture \(OLE Object\)](#)  
[RevertToSaved](#)

## **S**

[Save](#)  
[ScrollPage](#)  
[ScrollPosition](#)  
[Select](#)  
[SelectShapeType](#)  
[SendMail](#)  
[SetDefaults](#)  
[SetProtection](#)  
[ShowAll](#)  
[Spelling](#)

## **T**

[TileCharts](#)  
[ToBack \(Object object\)](#)  
[ToBack \(Chart object\)](#)  
[ToFront \(Object object\)](#)  
[ToFront \(Chart object\)](#)

## U

[UnattachFromLine](#)

[Undo](#)

[UnRegisterEvent](#)

[UpdateDateAndTime](#)

[UpdateFields](#)

## Related Topics

Description

[Language Reference](#)

## Objects, Alphabetical

Related

[Application](#)

[Chart](#)

[Charts collection](#)

[FieldTemplate](#)

[FieldTemplates collection](#)

[FieldValue](#)

[FieldValues collection](#)

[Font](#)

[Line](#)

[MasterItems](#)

[Menu collection](#)

[MenuItem](#)

[Object](#)

[Objects collection](#)

[OLE](#)

[PageLayout](#)

[Preferences](#)

[Shape](#)

[TextBlock](#)

## Related Topics

### Description

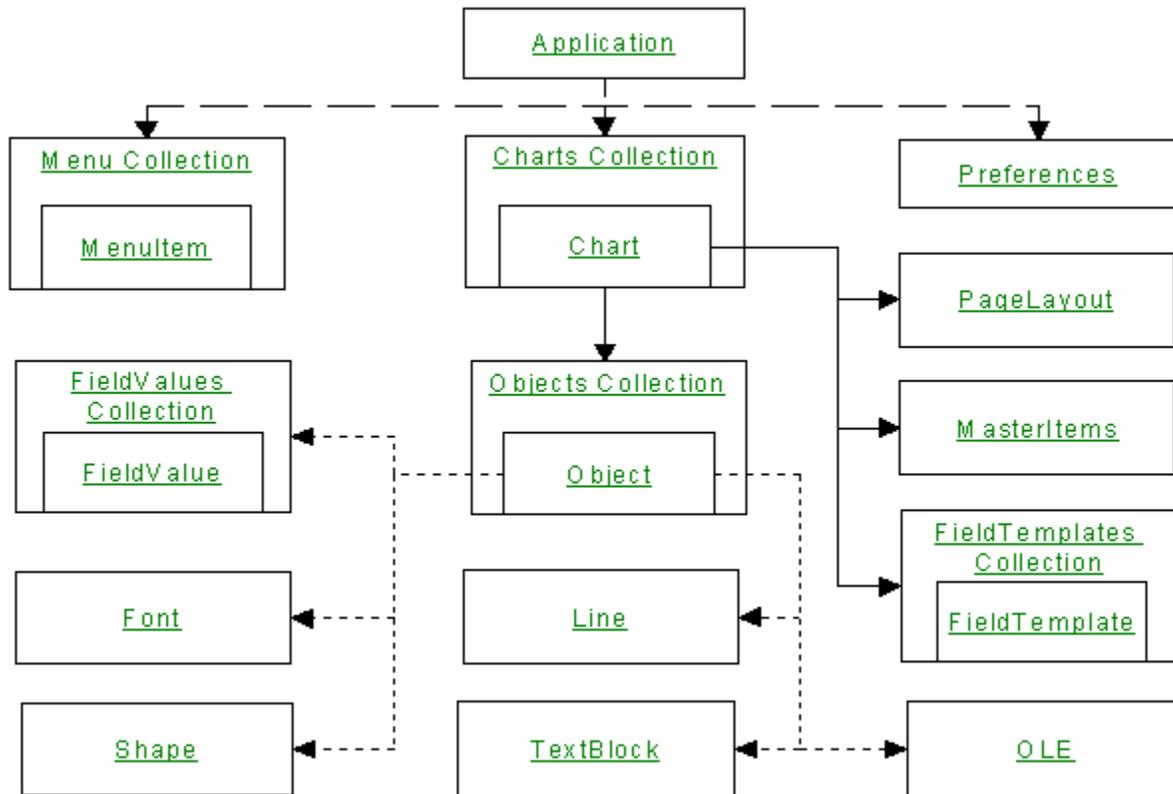
[Objects, graphical](#)

[Language Reference](#)

## Objects, Graphical

Related

The ABC OLE Automation object hierarchy, diagrammed below, shows how the objects and collections available in ABC OLE Automation relate to each other. The diagram includes the collections, which can contain more than one object. You can click on an object or collection to see the properties and methods contained in it and get information about those language elements.



At the top of the ABC OLE Automation hierarchy is the Application object, which is the interface to ABC. There can be only one ABC Application object at a time running on your system.

Branching from the Application object are the Charts collection, Preferences object, and Menus collection. An ABC Application can have multiple Charts collections and Menus collections, but only one Preferences object.

Branching from the Charts collection are the Chart objects. Each Chart object is restricted to a single PageLayout and MasterItems object, but can have multiple FieldTemplate and Object objects.

Below the Object object are the Shape, Line\_, TextBlock, OLE, and Font objects, and the FieldValues collection. Each Object object can have multiple FieldValue objects, but only one Shape, Line\_, TextBlock, OLE, and Font object.

## Related Topics

### Description

[Introducing ABC OLE Automation](#)

[Language Reference](#)

[Objects, alphabetical](#)

## Events

Related

A

B

C

Related

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

### A

[AppQuitSUBCLASS](#)

[AppMenuSUBCLASS](#)

[AppMenuHintSUBCLASS](#)

[AppMenuPopupSUBCLASS](#)

### C

[ChartActivateNOTIFY](#)

[ChartChangeNOTIFY](#)

[ChartCloseSUBCLASS](#)

[ChartNewNOTIFY](#)

[ChartOpenNOTIFY](#)

[ChartPasteNOTIFY](#)

### D

[DeleteSUBCLASS](#)

[DoubleClickSUBCLASS](#)

### E

[ExclusiveSelectionNOTIFY](#)

## **F**

FieldValueChangedNOTIFY

## **L**

LinkNOTIFY

## **N**

NewLineNOTIFY

NewShapeNOTIFY

## **O**

ObjectClickSUBCLASS

ObjectFontChangeNOTIFY

ObjectLineAttachNOTIFY

ObjectMovedNOTIFY

ObjectMoveSUBCLASS

ObjectSizedNOTIFY

ObjectSizeSUBCLASS

ObjectTextChangedNOTIFY

## **R**

ReplaceShapeNOTIFY

## **S**

SpecialKeySUBCLASS

## Related Topics

Description

[Language Reference](#)

## All Properties, Methods, Objects, and Events

Related

### A

[Accumulation property](#)

[AccumulationMethod property](#)

[Activate method \(Application object\)](#)

[Activate method \(Chart object\)](#)

[ActiveChart property](#)

[Add method \(Charts collection\)](#)

[Add method \(FieldTemplates collection\)](#)

[AddFromTemplate method](#)

[AddHorizontalGuideline method](#)

[AddMenu method](#)

[AddVerticalGuideline method](#)

[AlignToRulers property](#)

[AppendItem method](#)  
[Application object](#)  
[Application property](#)  
[AppMenuHintSUBCLASS event](#)  
[AppMenuPopupSUBCLASS event](#)  
[AppMenuSUBCLASS event](#)  
[AppQuitSUBCLASS event](#)  
[ArrangeIcons method](#)  
[AttachedToLine property](#)

## **B**

[BasicColor method](#)  
[Bold property](#)  
[BorderColor property](#)  
[BorderStyle property](#)  
[BorderWidth property](#)  
[Bottom property \(Object object\)](#)  
[Bottom property \(Application object\)](#)

## **C**

[CancelFullScreen method](#)  
[Caption property](#)  
[CascadeCharts method](#)  
[CenterX property](#)  
[CenterY property](#)  
[ChannelAlignment property](#)  
[Chart object](#)  
[ChartActivateNOTIFY event](#)  
[ChartChangeNOTIFY event](#)  
[ChartCloseSUBCLASS event](#)  
[ChartName property](#)  
[ChartNameShown property](#)  
[ChartNewNOTIFY event](#)  
[ChartOpenNOTIFY event](#)  
[ChartPasteNOTIFY event](#)  
[Charts collection](#)  
[Charts property](#)  
[ChartTypeShutdown method](#)  
[Checked property](#)  
[Clear method](#)  
[ClearGuidelines method](#)  
[ClipboardFormatAvailable property](#)  
[CloseAll method](#)  
[CloseChart method](#)  
[Color property \(Object object\)](#)  
[Color property \(Font object\)](#)  
[Color property \(Line\\_ object\)](#)  
[Copy method](#)  
[Count property](#)  
[CreateAddOn method](#)  
[CurrentLineRouting property](#)  
[CurrentShape property](#)  
[CurrentShapePalette property](#)  
[Cut method](#)

## **D**

[Date property](#)

[DateShown property](#)  
[DateStyle property](#)  
[DefaultFilePath property](#)  
[DeleteAll method](#)  
[DeleteField method](#)  
[DeleteItem method](#)  
[DeleteLines method](#)  
[DeleteSUBCLASS event](#)  
[DeselectAll method](#)  
[DestArrowColor property](#)  
[DestArrowSize property](#)  
[DestArrowStyle property](#)  
[Destination property](#)  
[DestinationDirection property](#)  
[DoubleClickSUBCLASS event](#)  
[DoVerb method](#)  
[DrawDirection property](#)  
[DrawFreeLine method](#)  
[DrawLine method](#)  
[DrawLineToOneObject method](#)  
[DrawPositionX property](#)  
[DrawPositionY property](#)  
[DrawShape method](#)  
[DrawSpacingX property](#)  
[DrawSpacingY property](#)  
[DrawTextBlock method](#)  
[Duplicate method \(Object object\)](#)  
[Duplicate method \(Chart object\)](#)

## **E**

[Empty method](#)  
[Enabled property](#)  
[ExclusiveSelectionNOTIFY event](#)

## **F**

[FieldFont property](#)  
[FieldNamesHidden property](#)  
[FieldPlacement property](#)  
[FieldsDaysPerWeek property](#)  
[FieldsHoursPerDay property](#)  
[FieldsOpaque property](#)  
[FieldTemplate object](#)  
[FieldTemplate property](#)  
[FieldTemplates collection](#)  
[FieldTemplates property](#)  
[FieldValue object](#)  
[FieldValueChangedNOTIFY event](#)  
[FieldValues collection](#)  
[FieldValues property](#)  
[FieldViewerVisible property](#)  
[FieldViewerWindowHandle property](#)  
[FillColor property](#)  
[FillPattern property](#)  
[FitShapeToText method](#)  
[Font object](#)  
[Font property](#)  
[Format property](#)

[FormattedValue property](#)  
[FullName property \(Application object\)](#)  
[FullName property \(Chart object\)](#)  
[FullScreen method](#)

## G

[GroupAndLink method](#)  
[GuidelinesOn property](#)

## H

[HasDiskFile property](#)  
[Height property \(Object object\)](#)  
[Height property \(Application object\)](#)  
[Height property \(PageLayout object\)](#)  
[Help method](#)  
[Hidden property](#)  
[HideAll method](#)  
[HidePercentGauge method](#)  
[Hint method](#)  
[Hourglass property](#)

## I

[IndexVisible property](#)  
[IndexWindowHandle property](#)  
[InsertItem method](#)  
[InsertObjectFromFile method](#)  
[IsEmpty property](#)  
[IsLaunched property](#)  
[IsLinked property](#)  
[Italic property](#)  
[Item method \(Objects collection\)](#)  
[Item method \(Charts collection\)](#)  
[Item method \(FieldTemplates collection\)](#)  
[Item method \(FieldValues collection\)](#)  
[Item method \(Menu collection\)](#)  
[ItemFromAll method](#)  
[ItemFromAttachments method](#)  
[ItemFromFieldValue method](#)  
[ItemFromLines method](#)  
[ItemFromNumber method](#)  
[ItemFromSelection method](#)  
[ItemFromShapes method](#)  
[ItemFromText method](#)  
[ItemFromUniqueID method](#)

## L

[Launch method](#)  
[LaunchCommand property](#)  
[LaunchFlags property](#)  
[LaunchIndicator property](#)  
[LaunchShadow property](#)  
[LaunchStartDir property](#)  
[Left property \(Object object\)](#)  
[Left property \(Application object\)](#)  
[Line\\_\\_object](#)  
[Line\\_\\_property](#)  
[LineCrossoverSize property](#)

[LineCrossoverStyle property](#)  
[LineSpacingX property](#)  
[LineSpacingY property](#)  
[Link method](#)  
[LinkedChartName property](#)  
[LinkFields property](#)  
[LinkIndicator property](#)  
[LinkNOTIFY event](#)  
[LinkShadow property](#)  
[Logo property](#)  
[LogoPathname property](#)  
[LogoShown property](#)

## **M**

[MakeRGB method](#)  
[MarginBottom property](#)  
[MarginLeft property](#)  
[MarginRight property](#)  
[MarginTop property](#)  
[MasterItems object](#)  
[MasterItems property](#)  
[Maximize method \(Application object\)](#)  
[Maximize method \(Chart object\)](#)  
[Menu collection](#)  
[MenuItem object](#)  
[Minimize method \(Application object\)](#)  
[Minimize method \(Chart object\)](#)  
[MsgBox method](#)

## **N**

[Name property \(Application object\)](#)  
[Name property \(Chart object\)](#)  
[Name property \(FieldTemplate object\)](#)  
[Name property \(FieldValue object\)](#)  
[Name property \(Font object\)](#)  
[New method](#)  
[NewFromTemplate method](#)  
[NewLineNOTIFY event](#)  
[NewShapeNOTIFY event](#)  
[NextNumber property](#)  
[NextShapeHeight property](#)  
[NextShapeWidth property](#)  
[NoRepaint property](#)  
[NoteFont property](#)  
[NoteIndicator property](#)  
[NoteShadow property](#)  
[NoteText property](#)  
[NoteTextLF](#)  
[NoteViewerVisible property](#)  
[NoteViewerWindowHandle property](#)  
[Number property](#)  
[NumberFont property](#)  
[NumberShown property](#)

## **O**

[Objects collection](#)  
[Object object](#)

[ObjectClickSUBCLASS event](#)  
[ObjectFontChangeNOTIFY event](#)  
[ObjectLineAttachNOTIFY event](#)  
[ObjectMovedNOTIFY event](#)  
[ObjectMoveSUBCLASS event](#)  
[Objects property](#)  
[ObjectSizedNOTIFY event](#)  
[ObjectSizeSUBCLASS event](#)  
[ObjectTextChangedNOTIFY event](#)  
[ObjectType property](#)  
[OLE object](#)  
[OLE property](#)  
[Opaque property](#)  
[Open method](#)  
[OperatingSystem property](#)  
[Orientation property](#)

## **P**

[PageCount property](#)  
[PageHeight property](#)  
[PageLayout object](#)  
[PageLayout property](#)  
[PageNumber property](#)  
[PageNumberShown property](#)  
[PageOrder property](#)  
[PageWidth property](#)  
[PaperSize property](#)  
[Parent property](#)  
[Paste method](#)  
[PasteLink method](#)  
[PasteSpecial method](#)  
[Path property](#)  
[PercentGauge method](#)  
[PercentGaugeCancelled method](#)  
[PercentGaugeValue property](#)  
[Preferences object](#)  
[Preferences property](#)  
[PrintBlankPages property](#)  
[Printer property](#)  
[PrintOut method](#)  
[PrintSelected method](#)  
[Protected property](#)

## **Q**

[Quit method](#)

## **R**

[Range property](#)  
[ReadOnly property](#)  
[RegisterEvent method](#)  
[RemoveAddOn method](#)  
[RemoveMenu method](#)  
[Renumber method](#)  
[Repaint method](#)  
[ReplaceShape method](#)  
[ReplaceShapeNOTIFY event](#)  
[Restore method \(Application object\)](#)

[Restore method \(Chart object\)](#)  
[RestorePicture method \(Object object\)](#)  
[RestorePicture method \(OLE object\)](#)  
[RevertToSaved method](#)  
[Right property \(Object object\)](#)  
[Right property \(Application object\)](#)

## **S**

[Save method](#)  
[Saved property](#)  
[ScrollLeft property](#)  
[ScrollPage method](#)  
[ScrollPosition method](#)  
[ScrollTop property](#)  
[Select method](#)  
[Selected property](#)  
[SelectedLineCount property](#)  
[SelectedObjectCount property](#)  
[SelectedOtherCount property](#)  
[SelectedShapeCount property](#)  
[SelectShapeType method](#)  
[SendMail method](#)  
[SetDefaults method](#)  
[SetProtection method](#)  
[ShadowColor property](#)  
[ShadowOffset property](#)  
[ShadowStyle property](#)  
[Shape object](#)  
[Shape property](#)  
[ShapeName property](#)  
[ShapePaletteVisible property](#)  
[ShapePaletteWindowHandle property](#)  
[ShapeSizing property](#)  
[ShowAll method](#)  
[ShowLegend property](#)  
[ShowNodesOnLines property](#)  
[ShowRulers property](#)  
[Size property](#)  
[SmartShapeSpacing property](#)  
[Source property](#)  
[SourceArrowColor property](#)  
[SourceArrowSize property](#)  
[SourceArrowStyle property](#)  
[SourceDirection](#)  
[SpecialKeySUBCLASS event](#)  
[Spelling method](#)  
[SSSHorizontal property](#)  
[SSSVertical property](#)  
[StatusBar property](#)  
[StemColor property](#)  
[StemStyle property](#)  
[StemWidth property](#)  
[StretchType property](#)  
[Strikethrough property](#)

## **T**

[Text property \(Object object\)](#)

[Text property \(Menu collection\)](#)  
[Text property \(MenuItem object\)](#)  
[Text1 property](#)  
[Text1Shown property](#)  
[Text2 property](#)  
[Text2Shown property](#)  
[TextAlignment property](#)  
[TextBlock object](#)  
[TextBlock property](#)  
[TextLF](#)  
[TileCharts method](#)  
[Time property](#)  
[TimeShown property](#)  
[ToBack method \(Object object\)](#)  
[ToBack method \(Chart object\)](#)  
[ToFront method \(Object object\)](#)  
[ToFront method \(Chart object\)](#)  
[Top property \(Object object\)](#)  
[Top property \(Application object\)](#)  
[TouchAlignment property](#)  
[Type property \(Object object\)](#)  
[Type property \(Chart object\)](#)  
[Type property \(FieldTemplate object\)](#)  
[Type property \(FieldValue object\)](#)  
[Type property \(Line object\)](#)  
[TypeRequiresEXE property](#)  
[TypeUsesEXE property](#)

## **U**

[UnattachFromLine method](#)  
[Underline property](#)  
[Undo method](#)  
[UndoAvailable property](#)  
[UniqueID property](#)  
[Units property \(Chart object\)](#)  
[Units property \(Preferences object\)](#)  
[UnRegisterEvent method](#)  
[UpdateDateAndTime method](#)  
[UpdateFields method](#)

## **V**

[Valid property](#)  
[Value property](#)  
[Version property](#)  
[View property](#)  
[Visible property \(Application object\)](#)  
[Visible property \(Menu collection\)](#)

## **W**

[Width property \(Object object\)](#)  
[Width property \(Application object\)](#)  
[Width property \(PageLayout object\)](#)  
[WindowHandle property](#)

## **Z**

[ZoomPercentage property](#)

## Related Topics

Description

[Language Reference](#)

## ABC Menu Command Equivalents

Related

This topic displays the menu commands available in ABC FlowCharter and the related OLE Automation Tool elements. When using an ABC OLE Automation Tool element, it is equivalent to executing the related command.

<b>File Menu Command</b>	<b>ABC OLE Automation Equivalent</b>
New	<a href="#">New method</a>
New From Template	<a href="#">AddFromTemplate method</a>
Open	<a href="#">Open method</a>
Close	<a href="#">CloseChart method</a>
CloseAll	<a href="#">CloseAll method</a>
Save	<a href="#">Save method</a>
Save As	<a href="#">Save method</a>
Page Layout	<a href="#">PageLayout object</a>
Print	<a href="#">PrintOut method</a>
	<a href="#">PrintSelected method</a>
Preferences	<a href="#">Preferences object</a>
Send Mail	<a href="#">SendMail method</a>
Exit	<a href="#">Quit method</a>
<b>Edit Menu Command</b>	<b>ABC OLE Automation Equivalent</b>
Undo	<a href="#">Undo method</a>
Cut	<a href="#">Cut method</a>
Copy	<a href="#">Copy method</a>
Paste	<a href="#">Paste method</a>
Paste Special	<a href="#">PasteSpecial method</a>
Paste Link	<a href="#">PasteLink method</a>
Clear	<a href="#">Clear method</a>
Select	<a href="#">Select method</a>
Restore Picture	<a href="#">RestorePicture method</a>
Spelling	<a href="#">Spelling method</a>
Insert Object	<a href="#">InsertObjectFromFile method</a>
Object	<a href="#">DoVerb method</a>
<b>Fields Menu Command</b>	<b>ABC OLE Automation Equivalent</b>
Setup	<a href="#">FieldTemplates collection</a>
Preferences	<a href="#">Preferences object</a>
Update	<a href="#">UpdateFields method</a>
Show Viewer	<a href="#">FieldViewerVisible property</a>
Show Legend	<a href="#">ShowLegend property</a>
<b>Window Menu Command</b>	<b>ABC OLE Automation Equivalent</b>
Tile	<a href="#">TileCharts method</a>
Cascade	<a href="#">CascadeCharts method</a>
Arrange Icons	<a href="#">Arrangelcons method</a>
Chart1	<a href="#">Activate method</a>
Full Screen	<a href="#">FullScreen method</a>

## Related Topics

Description

[Language Reference](#)

## Using C++ with ABC OLE Automation

Related

[C++ Sample  
Using C++](#)

## Related Topics

### Description

[Contents](#)

[Language Reference](#)

## C++ Sample

Related

A sample C++/MFC program named OLE\_VBX.EXE is included with ABC OLE Automation. This sample illustrates using ABC OLE Automation with C++. The OLE\_VBX.EXE file is automatically installed in the ABC\SAMPLES\OLE\_VBX directory by the installation program.

## Related Topics

### Description

[Using C++ with ABC OLE Automation](#)

## Using C++

Related

Use the Microsoft C++ ClassWizard to help you access ABC OLE Automation functionality. The OLE Automation Tab of the ClassWizard lets you read a typelib file and create a .H and a .CPP file from the typelib file. Use this function to create the ABC OLE Automation ABCAUTO.H and ABCAUTO.CPP files from the ABC typelib file named ABCAUTO.TLB.

After you have generated the ABCAUTO.CPP and ABCAUTO.H files, you can start making calls to ABC with a C++ program.

[C++: Creating an Application Object](#)

[C++: Events](#)

[C++: Notes](#)

## Related Topics

### Description

[Using C++ with ABC OLE Automation](#)

## C++: Creating an Application Object

Related

The first step in accessing ABC is to create an application object, as illustrated below.

```
#include "ABCAUTO.H"           // This file generated by ClassWizard

ABCFlowApp ABC;

ABC.CreateDispatch ("ABCFlow.application");
ABC.SetVisible (TRUE);
```

In the sample code above, type ABCFlowApp is a class generated by the ClassWizard defined in ABCAUTO.H. All of the ABC Application APIs are available in the ABCFlowApp class. All ABC properties are preceded with Set or Get, as explained in ABCAUTO.H. The call CreateDispatch("ABCFlow.application") performs the same function as Set ABC = CreateObject("ABCFlow.application") in Visual Basic. This call starts ABC if ABC is not already running, and returns a valid application object that you can use to call the APIs. The last line in this sample, ABC.SetVisible(TRUE), performs the same function as ABC.Visible = True in Visual Basic.

All other ABC Automation objects use AttachDispatch instead of CreateDispatch, as illustrated by the sample below.

```
// Setup the ABCChart
Chart ABCChart;

// Get the active Chart
ABCChart.AttachDispatch (ABC.GetActiveChart ());

VARIANT vEmpty;
VariantInit (&vEmpty);

ABCChart.DrawShape (vEmpty);
VariantClear (&vEmpty);
```

The sample code above assumes that ABC is a valid ABC application object. The call to ABCChart.AttachDispatch(ABC.GetActiveChart( )) puts the ABCChart object in a state that enables calls to all Chart Object APIs, such as the DrawShape call shown in the sample.

Chart.DrawShape takes one optional parameter, the name of the type of shape to draw. In Visual Basic, optional parameters are handled internally, so you can just omit an optional parameter and Visual Basic does all the work. However, in C++, you must declare a variable of type VARIANT, initialize it, and pass it to DrawShape. If you want to tell ABC to draw a Process shape, use the code shown below instead.

```
VARIANT vName;
VariantInit (&vName);

V_VT (&vName) = VT_BSTR;
V_BSTR (&vName) = SysAllocString ("Process");
ABCChart.DrawShape (vName);
VariantClear (&vName);
```

For more information on VARIANT variables, see the OLESDKV2.HLP file shipped with VC++ 1.5, and search for "Variant Manipulation Functions."

## Related Topics

### Description

[Using C++](#)

## C++: Events

Related

To use the ABC Events VBX in C++, you need to use the CVBControl interface provided in MFC. The simplest way is to check "Custom VBX Controls" on the Options menu of MFC AppWizard when starting your new project.

In the ABC OLE Automation sample OLE\_VBX.EXE program, the ABC Events VBX was dropped into the project's ABOUT box using AppStudio. Then the Message Maps Tab was chosen from the ClassWizard on the Resource menu. On the WM\_INITDIALOG message the events are registered with ABC.

```
LPDISPATCH lpVBXDisp =
(LPDISPATCH)m_pABCVBX->GetNumProperty("lVBX");

VARIANT vEmpty;

VariantInit(&vEmpty);
m_ABC.RegisterEvent(lpVBXDisp, "C++ Events Sample",
"DoubleClickSUBCLASS", vEmpty);
m_ABC.RegisterEvent(lpVBXDisp, "C++ Events Sample",
"DeleteSUBCLASS", vEmpty);
```

In the example above, note the use of *casting* on the lpVBXDisp assignment statement. MFC's VBX support does not allow the transfer of OLE IDispatch pointers to or from properties of the VBX. Therefore, a series of mirror invisible properties were added to the ABC VBX.

To allow the ABCAUTO.VBX to work with MFC, some redundant properties were added that are "Num" (long) properties. You **must** cast them to (LPDISPATCH) before using them.

ABC	C++
App	IApp
Chart	IChart
Object	IObject
Object2	IObject2
FieldValue	IFieldValue
Menu	IMenu
MenuItem	IMenuItem
VBX	IVBX

When responding to a VBX event, use the Message Maps Tab in App Studio to help link your code to the VBX. The sample below illustrates how a program can respond to the double click event from ABC.

```
Void cAboutDlg::OnDoubleClickSubclassAbc1(Unit, int, Cwnd*, LPVOID)
{
    // Setup the ABCObject
    Object ABCObject

    ABCObject.AttachDispatch((LPDISPATCH)m_pABCVBX->
        GetNumProperty("lObject"), FALSE);

    // Set the passed object to green
    ABCObject.SetColor(RGB(0,0xff,0));

    ABCObject.SetText("C++ is easy");

    m_pABCVBX->SetNumProperty("Override", TRUE);
}
```

## Related Topics

### Description

[Using C++](#)

## C++: Notes

Related

When using methods and properties that return a string, check to see if the return value is NULL before assigning it to a CString.

Additionally, all strings returned from ABC OLE Automation must be freed using SysFreeString.

## Related Topics

### Description

[Using C++](#)

## Application Object

**Description** The Application object is at the top of the ABC OLE Automation hierarchy. It is the interface to ABC OLE Automation. There can be only one Application object at a time running in your system. Below the ABC Application object are the Charts collection, Preferences object, and Menus collection. You can have multiple Charts collections and Menus collections, but only one Preferences object.

### Properties

---

[ActiveChart](#)  
[Application](#)  
[Bottom](#)  
[Caption](#)  
[Charts](#)  
[DefaultFilePath](#)  
[FieldViewerVisible](#)  
[FieldViewerWindowHandle](#)  
[FullName](#)  
[Height](#)  
[Hourglass](#)  
[IndexVisible](#)  
[IndexWindowHandle](#)  
[Left](#)  
[Name](#)  
[NoteViewerVisible](#)  
[NoteViewerWindowHandle](#)  
[OperatingSystem](#)  
[Parent](#)  
[Path](#)  
[PercentGaugeValue](#)  
[Preferences](#)  
[Printer](#)  
[Right](#)  
[ShapePaletteVisible](#)  
[ShapePaletteWindowHandle](#)  
[StatusBar](#)  
[Top](#)  
[UndoAvailable](#)  
[Version](#)  
[Visible](#)  
[Width](#)  
[WindowHandle](#)

### Methods

---

[Activate](#)  
[AddMenu](#)  
[Arrangelcons](#)  
[BasicColor](#)  
[CascadeCharts](#)  
[ChartTypeShutdown](#)  
[CloseAll](#)  
[CreateAddOn](#)  
[Help](#)  
[HidePercentGauge](#)  
[Hint](#)  
[MakeRGB](#)  
[Maximize](#)  
[Minimize](#)  
[MsgBox](#)  
[New](#)  
[NewFromTemplate](#)  
[Open](#)  
[Quit](#)  
[PercentGauge](#)  
[PercentGaugeCancelled](#)  
[RegisterEvent](#)  
[RemoveAddOn](#)  
[RemoveMenu](#)  
[Restore](#)  
[TileCharts](#)  
[Undo](#)  
[UnRegisterEvent](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

[VBX Event Variables](#)

## ActiveChart Property

Related

Related

Related

**Usage** *ApplicationObject.ActiveChart*

**Description** You use the **ActiveChart** property to find the active Chart object in the Application. This is the simplest way to be sure that you are operating on the current chart. The **ActiveChart** property is read only.

**Data Type** Object

**Value** The currently active chart

**ABC Equivalent** None

## Related Topics

### Language Elements

[Activate Method \(Application Object\)](#)

[Name Property \(Application Object\)](#)

### Description

[Activating a Chart](#)

### Object

[Application Object](#)

## Bottom Property (Application Object)

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Bottom = PositionInPixels</i>
<b>Description</b>	The <b>Bottom</b> property lets you specify the position of the bottom of the ABC window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The <b>Bottom</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The number of pixels from the bottom of the screen to the bottom of the ABC window
<b>ABC Equivalent</b>	The <b>Bottom</b> property is equivalent to resizing the ABC window.

## Related Topics

### Language Elements

[Bottom Property \(Object Object\)](#)

[Height Property \(Application Object\)](#)

[Left Property \(Application Object\)](#)

[Right Property \(Application Object\)](#)

[Top Property \(Application Object\)](#)

[Width Property \(Application Object\)](#)

### Description

[Displaying the Field Viewer, Notes Viewer, and Shape Palette](#)

### Object

[Application Object](#)

## Caption Property

Related

Related

Related

**Usage** *ApplicationObject.Caption = Title*

**Description** The **Caption** property lets you customize ABC by changing what it says in the title bar. Set the **Caption** property to "" to restore the standard ABC caption ("Micrografx ABC FlowCharter"). The **Caption** property is read/write.

**Data Type** String

**Value** The text in the title bar of ABC

**ABC Equivalent** None

## Related Topics

### Description

[Changing the ABC Title Bar](#)

### Object

[Application Object](#)

## Charts Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Charts</i>
<b>Description</b>	The <b>Charts</b> property lets you find the charts included in the Charts collection. The <b>Charts</b> property is read only, but all the properties from the object it returns are read/write.
<b>Data Type</b>	Collection object
<b>Value</b>	The <b>Charts</b> property returns the charts included in the Charts collection.
<b>ABC Equivalent</b>	None

## Related Topics

### Description

[Identifying a Chart's Filename](#)

### Object

[Application Object](#)

**Close** **Copy** **Print**

## Charts Property Example

This example uses the **Charts** property of the Application object to put the chart collection into a variable.

```
Dim ABC As Object
Dim Application_Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

Set Application_Chart = ABC.Charts                     ' Set the collection of open ABC charts
```

## DefaultFilePath Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.DefaultFilePath = Path</i> The <i>Path</i> element is the default file path.
<b>Description</b>	You use the <b>DefaultFilePath</b> property to find or set the default path for all files that are opened or saved. The <b>DefaultFilePath</b> property is read/write.
<b>Data Type</b>	String
<b>Value</b>	The default file path
<b>ABC Equivalent</b>	None

## Related Topics

### Description

[Setting a Default Path for Charts](#)

### Object

[Application Object](#)

Close Copy Print

## Assorted Application Object Properties Example 1

This example uses properties of the Application object to find and display the default file path, path, operating system, current printer, and whether Undo is available.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Dim App_File_Path As String
App_File_Path = ABC.DefaultFilePath                   ' Get application default path
MsgBox "Application Default File Path is " + App_File_Path ' Display

Dim EXE_Path As String
EXE_Path = ABC.Path                                   ' Get path to ABC.EXE (executable file)
MsgBox "The ABC.EXE path is " + EXE_Path              ' Display

Dim Operating_System As String
Operating_System = ABC.OperatingSystem                ' Get operating system
MsgBox "ABC is running on " + Operating_System        ' Display

Dim ABC_Printer As String
ABC_Printer = ABC.Printer                             ' Get current printer
MsgBox "Current Printer is " + ABC_Printer            ' Display

Dim Undo_Status As Integer
Undo_Status = ABC.UndoAvailable                       ' Get undo status
Select Case Undo_Status                               ' Display
    Case True
        MsgBox "Undo is available."
    Case Else
        MsgBox "Undo is unavailable."
End Select
```

## FieldViewerVisible Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject</i> . <b>FieldViewerVisible</b> = {True   False}
<b>Description</b>	The <b>FieldViewerVisible</b> property lets you show or hide the ABC Field Viewer. The <b>FieldViewerVisible</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes the Field Viewer visible; False makes it invisible.
<b>ABC Equivalent</b>	The <b>FieldViewerVisible</b> property is equivalent to choosing Show Viewer or Hide Viewer in the ABC Field menu.

## Related Topics

### Language Elements

[IndexVisible Property](#)

[NoteViewerVisible Property](#)

[ShapePaletteVisible Property](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Displaying the Field Viewer, Notes Viewer, and Shape Palette](#)

[Opening the Field Viewer](#)

### Object

[Application Object](#)

Close Copy Print

## Assorted Application Object Properties Example 2

This example uses properties of the Application object to see windows belonging to ABC and find their window handles.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawPositionX = 2                               ' Set drawing position
Chart.DrawPositionY = 2                               ' (Default is inches)

Chart.DrawShape ("delay")                             ' Create a shape

Chart.FieldTemplates.Add("Inventory")                  ' Create a field for the shape

ABC.FieldViewerVisible = True                          ' Make field viewer visible
Dim Field_View Window_Handle As Long
Field_View Window_Handle = ABC.FieldViewerWindowHandle ' Get Window Handle
MsgBox "Field Viewer Window Handle = " + CStr(Field_View Window_Handle)
ABC.FieldViewerVisible = False                         ' Make field viewer invisible

ABC.NoteViewerVisible = True                           ' Make note viewer visible
Dim Note_View Window_Handle As Long
Note_View Window_Handle = ABC.NoteViewerWindowHandle  ' Get Window Handle
MsgBox "Note Viewer Window Handle = " + CStr(Note_View Window_Handle)
ABC.NoteViewerVisible = False                          ' Make note viewer invisible

ABC.IndexVisible = True                                ' Make index visible
Dim Index_Window_Handle As Long
Index_Window_Handle = ABC.IndexWindowHandle            ' Get Window Handle
MsgBox "Index Window Handle = " + CStr(Index_Window_Handle)
ABC.IndexVisible = False                               ' Make index invisible

ABC.ShapePaletteVisible = True                         ' Make shape palette visible
Dim Shape_Palette_Window_Handle As Long
Shape_Palette_Window_Handle = ABC.ShapePaletteWindowHandle ' Get Window Handle
MsgBox "Shape Palette Window Handle = " + CStr(Shape_Palette_Window_Handle)
ABC.ShapePaletteVisible = False                       ' Make shape palette invisible
```

## FieldViewerWindowHandle Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject</i> . <b>FieldViewerWindowHandle</b>
<b>Description</b>	The <b>FieldViewerWindowHandle</b> property lets you find the handle to the window of the Field Viewer. If the window is not visible, its value is Null. The <b>FieldViewerWindowHandle</b> property is read only.
<b>Data Type</b>	Long
<b>Value</b>	The handle to the Field Viewer window. If the window is not visible, the value is Null.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[IndexWindowHandle Property](#)

[NoteViewerWindowHandle Property](#)

[ShapePaletteWindowHandle Property](#)

[WindowHandle Property](#)

### Description

[Window Handles](#)

### Object

[Application Object](#)

## FullName Property (Application Object)

Related

Related

Related

**Usage** *ApplicationObject.FullName*

**Description** The **FullName** property of the Application object lets you find the ABC path, including the executable filename. To get the path without the executable file name, use the **Path** property. The **FullName** property is read only.

**Data Type** String

**Value** The fully qualified path of the ABC program that is running, including the name of the executable file

**ABC Equivalent** None

## Related Topics

### Language Elements

[FullName Property \(Chart Object\)](#)

[Name Property \(Application Object\)](#)

[OperatingSystem Property](#)

[Path Property](#)

[Version Property](#)

### Description

[Getting ABC System Information](#)

### Object

[Application Object](#)

[Close](#) [Copy](#) [Print](#)

## FullName Property and Name Property (Application Object) Example

This example uses the **FullName** property and **Name** property of the Application object to find and display the full name and name of the running application.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC

MsgBox "Application Full Name = " + ABC.FullName       ' Display

MsgBox "Default Application Name = " + ABC.Name        ' Display
```

## Height Property (Application Object)

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Height = HeightInPixels</i>
<b>Description</b>	The <b>Height</b> property lets you specify the position of the height of the ABC window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The <b>Height</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The height of the ABC window in pixels
<b>ABC Equivalent</b>	The <b>Height</b> property is equivalent to resizing the ABC window.

## Related Topics

### Language Elements

[Bottom Property \(Object Object\)](#)

[Height Property \(Object Object\)](#)

[Height Property \(PageLayout Object\)](#)

[Left Property \(Application Object\)](#)

[Right Property \(Application Object\)](#)

[Top Property \(Application Object\)](#)

[Width Property \(Application Object\)](#)

### Description

[Positioning and Resizing the ABC Window](#)

### Object

[Application Object](#)

Close Copy Print

## Height, Width Property (Application Object) Example

This example uses the **Height** property and **Width** property of the Application object to find and display the height and width of the ABC window.

```
Dim ABC As Object
Dim Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
Set Chart = ABC.ActiveChart                             ' Get the active chart

MsgBox "Application Height = " + ABC.Height            ' Display

MsgBox "Application Width = " + ABC.Width              ' Display
```

## IndexVisible Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.IndexVisible</i> = {True   False}
<b>Description</b>	The <b>IndexVisible</b> property lets you show or hide the ABC Index. The <b>IndexVisible</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes the Index visible; False makes it invisible.
<b>ABC Equivalent</b>	The <b>IndexVisible</b> property is equivalent to clicking the Index button at the bottom of the ABC window.

## Related Topics

### Language Elements

[FieldViewerVisible Property](#)

[NoteViewerVisible Property](#)

[ShapePaletteVisible Property](#)

### Description

[Displaying the Field Viewer, Notes Viewer, and Shape Palette](#)

### Object

[Application Object](#)

## IndexWindowHandle Property

Related

Related

Related

**Usage** *ApplicationObject.IndexWindowHandle*

**Description** The **IndexWindowHandle** property lets you find the handle to the window of the Index. If the window is not visible, its value is Null. The **IndexWindowHandle** property is read only.

**Data Type** Long

**Value** The handle to the Index window. If the window is not visible, the value is Null.

**ABC Equivalent** None

## Related Topics

### Language Elements

[FieldViewerWindowHandle Property](#)

[NoteViewerWindowHandle Property](#)

[ShapePaletteWindowHandle Property](#)

[WindowHandle Property](#)

### Description

[Window Handles](#)

### Object

[Application Object](#)

## Left Property (Application Object)

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Left = PositionInPixels</i>
<b>Description</b>	The <b>Left</b> property lets you specify the position of the left side of the ABC window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The <b>Left</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The number of pixels from the left of the screen to the left side of the ABC window
<b>ABC Equivalent</b>	The <b>Left</b> property is equivalent to resizing the ABC window.

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[Height Property \(Application Object\)](#)

[Left Property \(Object Object\)](#)

[Right Property \(Application Object\)](#)

[Top Property \(Application Object\)](#)

[Width Property \(Application Object\)](#)

### Description

[Positioning and Resizing the ABC Window](#)

### Object

[Application Object](#)

## Name Property (Application Object)

Related

Related

Related

**Usage** *ApplicationObject.Name*

**Description** The **Name** property always equals "ABC FlowCharter" for compatibility with all ABC products. The **Name** property is read only.

**Data Type** String

**Value** Always equals "ABC FlowCharter"

**ABC Equivalent** None

## Related Topics

### Language Elements

[FullName Property \(Application Object\)](#)

[Name Property \(Chart Object\)](#)

[Name Property \(FieldTemplate Object\)](#)

[Name Property \(FieldValue Object\)](#)

[Name Property \(Font Object\)](#)

[OperatingSystem Property](#)

[Path Property](#)

[Version Property](#)

### Description

[Getting ABC System Information](#)

### Object

[Application Object](#)

## NoteViewerVisible Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.NoteViewerVisible</i> = {True   False}
<b>Description</b>	You use the <b>NoteViewerVisible</b> property to find or set whether the Note window is open or closed. The <b>NoteViewerVisible</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes the Note Viewer visible; False makes it invisible.
<b>ABC Equivalent</b>	The <b>NoteViewerVisible</b> property is equivalent to clicking the Note button at the bottom of the ABC window.

## Related Topics

### Language Elements

[NoteIndicator Property](#)

[NoteShadow Property](#)

[NoteText Property](#)

### Description

[Opening the Note Window](#)

[Displaying the Field Viewer, Notes Viewer, and Shape Palette](#)

[Working with Notes](#)

### Object

[Application Object](#)

## NoteViewerWindowHandle Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject</i> . <b>NoteViewerWindowHandle</b>
<b>Description</b>	The <b>NoteViewerWindowHandle</b> property lets you find the handle to the window of the Note Viewer. If the window is not visible, its value is Null. The <b>NoteViewerWindowHandle</b> property is read only.
<b>Data Type</b>	Long
<b>Value</b>	The handle to the Note Viewer window. If the window is not visible, the value is Null.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FieldViewerWindowHandle Property](#)

[IndexWindowHandle Property](#)

[ShapePaletteWindowHandle Property](#)

[WindowHandle Property](#)

### Description

[Window Handles](#)

### Object

[Application Object](#)

## OperatingSystem Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.OperatingSystem</i>
<b>Description</b>	The <b>OperatingSystem</b> property lets you find the operating system under which ABC is running. The <b>OperatingSystem</b> property is read only.
<b>Data Type</b>	String
<b>Value</b>	The operating system under which the ABC program is running. For example, it equals "DOS 6.21;Windows 3.11" if you are running those versions.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FullName Property \(Application Object\)](#)

[Name Property \(Application Object\)](#)

[Path Property](#)

[Version Property](#)

### Description

[Getting ABC System Information](#)

### Object

[Application Object](#)

## Path Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Path</i>
<b>Description</b>	The <b>Path</b> property lets you find the ABC application path, excluding the executable filename. The path does not include a final back slash (\). To get the path with the executable file name, use the <b>FullName</b> property. The <b>Path</b> property is read only.
<b>Data Type</b>	String
<b>Value</b>	The fully qualified path of the ABC program that is running, excluding the name of the executable file
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FullName Property \(Application Object\)](#)

[Name Property \(Application Object\)](#)

[OperatingSystem Property](#)

[Version Property](#)

### Description

[Getting ABC System Information](#)

### Object

[Application Object](#)

## PercentGaugeValue Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.PercentGaugeValue = PercentageDone</i>
<b>Description</b>	The <b>PercentGaugeValue</b> property lets you set the value in the Percent Gauge dialog box you created using the <b>PercentGauge</b> method. The <b>PercentGaugeValue</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The value of the percent gauge
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[HidePercentGauge Method](#)

[Hint Method](#)

[Hourglass Property](#)

[MsgBox Method](#)

[PercentGauge Method](#)

[PercentGaugeCancelled Method](#)

[StatusBar Property](#)

### Description

[Providing Feedback](#)

### Object

[Application Object](#)

## Preferences Property

Related

Related

Related

**Usage** *ApplicationObject.Preferences*

**Description** The **Preferences** property lets you find the Preferences object. The **Preferences** property is read only, but all the properties from the object it returns are read/write.

**Data Type** Object

**Value** The Preferences object

**ABC Equivalent** None

## Related Topics

### Description

[Setting Preferences](#)

### Object

[Application Object](#)

Close Copy Print

## Preferences Property Example

This example uses the **Preferences** property of the Application object to put the preferences collection into a variable.

```
Dim ABC As Object
Dim App_Preferences As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True

Set App_Preferences = ABC.Preferences
```

' Start ABC  
' Make ABC visible  
' Set the collection of open ABC Charts

## Printer Property

Related

Related

Related

### Usage

*ApplicationObject.Printer = {PrinterName | PrinterPort}*

### Description

The **Printer** property lets you find or set the current printer. When you read the value of the **Printer** property, it returns the current printer and port. For example, it might return "HP LaserJet III on LPT2:." When you set the value, the program uses a "loose matching" routine that starts at the beginning of the string. For example, setting the Printer property to "HP Laser" or "LPT2" chooses "HP LaserJet III on LPT2:" if that is the printer on LPT2:. If more than one printer matches the value you set, the first one alphabetically is used.

### Data Type

String

### Value

The current printer

### ABC Equivalent

The **Printer** property is equivalent to choosing Printer Setup in the File menu and choosing the printer you want to use.

## **Related Topics**

### **Language Elements**

[PrintOut Method](#)

[PrintSelected Method](#)

### **Description**

[Printing Charts](#)

### **Object**

[Application Object](#)

## Right Property (Application Object)

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Right = PositionInPixels</i>
<b>Description</b>	The <b>Right</b> property lets you specify the position of the right side of the ABC window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The <b>Right</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The number of pixels from the right of the screen to the right side of the ABC window
<b>ABC Equivalent</b>	The <b>Right</b> property is equivalent to resizing the ABC window.

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[Height Property \(Application Object\)](#)

[Left Property \(Application Object\)](#)

[Right Property \(Object Object\)](#)

[Top Property \(Application Object\)](#)

[Width Property \(Application Object\)](#)

### Description

[Positioning and Resizing the ABC Window](#)

### Object

[Application Object](#)

## ShapePaletteVisible Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.ShapePaletteVisible</i> = {True   False}
<b>Description</b>	You use the <b>ShapePaletteVisible</b> property to find or set whether the Shape Palette window is open or closed. The <b>ShapePaletteVisible</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes the Shape Palette visible; False makes it invisible.
<b>ABC Equivalent</b>	The <b>ShapePaletteVisible</b> property is equivalent to clicking the Show/Hide Shape Palette button at the bottom of the screen.

## Related Topics

### Language Elements

[FieldViewerVisible Property](#)

[IndexVisible Property](#)

[NoteViewerVisible Property](#)

### Description

[Displaying the Field Viewer, Notes Viewer, and Shape Palette](#)

[Using the Shape Palette](#)

### Object

[Application Object](#)

## ShapePaletteWindowHandle Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject</i> . <b>ShapePaletteWindowHandle</b>
<b>Description</b>	The <b>ShapePaletteWindowHandle</b> property lets you find the handle to the window of the Shape Palette. If the window is not visible, its value is Null. The <b>ShapePaletteWindowHandle</b> property is read only.
<b>Data Type</b>	Long
<b>Value</b>	The handle to the Shape Palette window. If the window is not visible, the value is Null.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FieldViewerWindowHandle Property](#)

[IndexWindowHandle Property](#)

[NoteViewerWindowHandle Property](#)

[WindowHandle Property](#)

### Description

[Window Handles](#)

### Object

[Application Object](#)

## StatusBar Property

Related

Related

Related

**Usage** *ApplicationObject.StatusBar = StatusBarText*

**Description** The **StatusBar** property lets you customize ABC by changing what it says in the status bar. You can restore the normal status bar hints by setting the **StatusBar** property to "". To set a temporary message in the hint line, use the **Hint** method. The **StatusBar** property is read/write.

**Data Type** String

**Value** The text in the status bar

**ABC Equivalent** None

## Related Topics

### Language Elements

[Hint Method](#)

### Description

[Changing the ABC Status Bar](#)

### Object

[Application Object](#)

## Top Property (Application Object)

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Top = PositionInPixels</i>
<b>Description</b>	The <b>Top</b> property lets you specify the position of the top of the ABC window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The <b>Top</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The number of pixels from the top of the screen to the top of the ABC window
<b>ABC Equivalent</b>	The <b>Top</b> property is equivalent to resizing the ABC window.

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[Height Property \(Application Object\)](#)

[Left Property \(Application Object\)](#)

[Right Property \(Application Object\)](#)

[Top Property \(Object Object\)](#)

[Width Property \(Application Object\)](#)

### Description

[Positioning and Resizing the ABC Window](#)

### Object

[Application Object](#)

Close Copy Print

## Top Property (Application Object) Example

This example uses the **Top** property, **Bottom** property, **Left** property, and **Right** property of the Application object to find and display the location of the ABC window.

```
Dim ABC As Object
Dim Chart As Object
Dim App_Top As Long, App_Bottom As Long, App_Left As Long, App_Right As Long

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

App_Top = ABC.application.Top
App_Bottom = ABC.application.Bottom
App_Left = ABC.application.Left
App_Right = ABC.application.Right

MsgBox "ABC's window border location is: Top = " + CStr(App_Top) + ", Bottom = " +
CStr(App_Bottom) + ", Left = " + CStr(App_Left) + ", and Right = " + CStr(App_Right)
```

## UndoAvailable Property

Related

Example1

Example2

Related

### Usage

*ApplicationObject*.UndoAvailable

### Description

The **UndoAvailable** property lets you find if there is anything to undo. The **UndoAvailable** property is read only.

### Data Type

Integer (Boolean)

### Value

True means something is available to undo; False means nothing is available to undo.

### ABC Equivalent

None

## **Related Topics**

**Language Elements**

[Undo Method](#)

**Description**

[Undoing Actions](#)

**Object**

[Application Object](#)

## Version Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Version</i>
<b>Description</b>	The <b>Version</b> property lets you find the version of the ABC OLE automation application object that is running. Note that this is not the version number of the ABC application, but rather the version number of the ABC OLE Automation API set. For example, for ABC FlowCharter 4.0, the <b>Version</b> property returns 1.0. The <b>Version</b> property is read only.
<b>Data Type</b>	String
<b>Value</b>	The version of the ABC OLE Automation application object that is running. For example, it equals "1.0" if you are running ABC FlowCharter 4.0 or ABC Viewer 1.0.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FullName Property \(Application Object\)](#)

[Name Property \(Application Object\)](#)

[OperatingSystem Property](#)

[Path Property](#)

### Description

[Getting ABC System Information](#)

### Object

[Application Object](#)

Close Copy Print

## Object Object Properties Example

This example uses properties of the Object object to determine the application's version, if the application is visible, what the application's window handle is, the message in the application's status bar, and the application's caption in the title bar.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
Set Chart = ABC.ActiveChart                             ' Get the active chart

Dim App_Version As String                               ' Get application version
App_Version = ABC.Version
MsgBox "Application version = " + App_Version           ' Display

Dim App_Visible As Integer                             ' Get application visibility state
App_Visible = ABC.Visible
Select Case App_Visible                                ' Display
    Case True
        MsgBox "Application is visible."
    Case Else
        MsgBox "Application is not visible."
End Select

Dim App_Window_Handle As Long                          ' Get application window handle
App_Window_Handle = ABC.WindowHandle
MsgBox "Application window handle = " + CStr(App_Window_Handle) ' Display

Dim App_Status_Bar As String                           ' Get application status bar message
App_Status_Bar = ABC.StatusBar
MsgBox "Application status bar = " + CStr(App_Status_Bar) ' Display

Dim App_Caption As String                              ' Get application caption
App_Caption = ABC.Caption
MsgBox "Application caption = " + App_Caption           ' Display
```

## Visible Property (Application Object)

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Visible</i> = {True   False}
<b>Description</b>	If you set the <b>Visible</b> property to True, the application is visible. If you set the <b>Visible</b> property to False, the application is still running, but it is not visible. You cannot switch to it using <b>Alt+Tab</b> , and it is not shown in the Task List dialog box that appears when you press <b>Ctrl+Esc</b> . The value False is the default, so you must begin all your programs by setting it to True. The <b>Visible</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes ABC visible. False makes ABC not visible; you cannot switch to it using <b>Alt+Tab</b> , and it is not shown in the Task List dialog box that appears when you press <b>Ctrl+Esc</b> .
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Activate Method \(Application Object\)](#)

[Activate Method \(Chart Object\)](#)

### Description

[Starting ABC](#)

### Object

[Application Object](#)

## Width Property (Application Object)

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Width = WidthInPixels</i>
<b>Description</b>	The <b>Width</b> property lets you specify the position of the width of the ABC window in pixels. The number of pixels available depends on your screen resolution. For example, if you are running standard VGA, your screen is 640 pixels wide and 480 pixels high. The <b>Width</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The width of the ABC window in pixels
<b>ABC Equivalent</b>	The <b>Width</b> property is equivalent to resizing the ABC window.

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[Height Property \(Application Object\)](#)

[Left Property \(Application Object\)](#)

[Right Property \(Application Object\)](#)

[Top Property \(Application Object\)](#)

[Width Property \(Object Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Positioning and Resizing the ABC Window](#)

### Object

[Application Object](#)

## WindowHandle Property Related

Related

Related

<b>Usage</b>	<i>ApplicationObject</i> . <b>WindowHandle</b> <i>ChartObject</i> . <b>WindowHandle</b>
<b>Description</b>	The <b>WindowHandle</b> property lets you find the handle to the window of ABC or of a chart. If the window is not visible, its value is Null. The <b>WindowHandle</b> property is read only.
<b>Data Type</b>	Long
<b>Value</b>	The handle to the window of ABC or the chart. If the window is not visible, the value is Null.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FieldViewerWindowHandle Property](#)

[IndexWindowHandle Property](#)

[NoteViewerWindowHandle Property](#)

[ShapePaletteWindowHandle Property](#)

### Description

[Window Handles](#)

### Object

[Application Object](#)

Close Copy Print

## WindowHandle Property Example

This example uses the **WindowHandle** property of the Chart object to find a chart's window handle. Also see the [Object Object Properties](#) example.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

MsgBox "The window handle for this chart is " + Chart.WindowHandle + "."
```

## Activate Method (Application Object)

Related

Related

Related

**Usage** *ApplicationObject.Activate*

**Description** You bring ABC to the front using the **Activate** method of the Application object. You usually have to do this only after the user has done something that moves ABC to the back, such as clicking another application that is visible on the screen or switching to another application using **Alt+Tab** or **Ctrl+Esc**.

**ABC Equivalent** The **Activate** method is equivalent to clicking the ABC window or pressing **Alt+Tab** or **Ctrl+Esc** to bring ABC to the front.

## Related Topics

### Language Elements

[Activate Method \(Chart Object\)](#)

[Visible Property \(Application Object\)](#)

### Description

[Bringing ABC or a Chart to the Front](#)

### Object

[Application Object](#)

**Related**

## Activate Method (Application Object) Example

This example uses the **Activate** method of the Application object to activate ABC.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True

ABC.Activate
```

' Start ABC  
' Make ABC visible  
' Activate ABC

## Arrangelcons Method

Related

Related

Related

**Usage** *ApplicationObject.Arrangelcons*

**Description** When you have several ABC chart windows minimized to icons, you can arrange them at the bottom of the ABC window using the **Arrangelcons** method.

**ABC Equivalent** The **Arrangelcons** method is equivalent to choosing Arrange Icons in the ABC Window menu.

## Related Topics

### Description

[Arranging ABC Icons](#)

### Object

[Application Object](#)

Related

## Arrangelcons Method Example

This example uses the **Arrangelcons** method of the Application object to arrange ABC icons. For this call to have any visible effect, one or more chart windows must be minimized and be moved from their original positions.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True

ABC.ArrangeIcons
```

```
' Start ABC
' Make ABC visible
' Arrange icons
```

## BasicColor Method

Related

Related

Related

### Usage

*ApplicationObject*.**BasicColor** (*Color*)

The *Color* element is an integer representing one of the sixteen standard VGA colors.

### Description

The **BasicColor** method lets you set colors from the sixteen VGA colors. The method returns the color as a long decimal value. You cannot change the values in the **BasicColor** method. For example, you cannot make `BasicColor(10)` yield the color purple.

### Data Type

Long

### Value

The **BasicColor** method returns one of the following values, based on the *Color* element.

<b>Color</b>	<b>BasicColor</b>	
<b>Element</b>	<b>Value</b>	<b>Result</b>
0	16777215	White
1	0	Black
2	255	Red
3	65280	Green
4	16711680	Blue
5	65535	Yellow
6	16711935	Magenta
7	16776960	Cyan
8	12632256	Gray
9	127	Dark Red
10	32512	Dark Green
11	8323072	Dark Blue
12	326397	Dark Yellow
13	8323199	Dark Magenta
14	8355584	Dark Cyan
15	8355711	Dark Gray

**ABC Equivalent** None

## Related Topics

### Language Elements

[MakeRGB Method](#)

### Description

[Color Constants Description](#)

[BasicColor Method description](#)

### Object

[Application Object](#)

**Related**

## BasicColor Method Example

This example uses the **BasicColor** method of the Application object to arrange ABC icons. For this call to function, ABC must be active and chart windows must be minimized.

```
Dim ABC As Object
Dim Basic_Color As Long
Dim User_Input As String

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                    ' Make ABC visible

' Get Basic Color for input value from 0 - 15
a:
User_Input = InputBox$("Please Enter a Windows color with a value from 0 - 15", "Basic Color")

Select Case User_Input
    Case "0"
        MsgBox "Entered color is WHITE."
    Case "1"
        MsgBox "Entered color is BLACK."
    Case "2"
        MsgBox "Entered color is RED."
    Case "3"
        MsgBox "Entered color is GREEN."
    Case "4"
        MsgBox "Entered color is BLUE."
    Case "5"
        MsgBox "Entered color is YELLOW."
    Case "6"
        MsgBox "Entered color is MAGENTA."
    Case "7"
        MsgBox "Entered color is CYAN."
    Case "8"
        MsgBox "Entered color is GRAY."
    Case "9"
        MsgBox "Entered color is DK_RED."
    Case "10"
        MsgBox "Entered color is DK_GREEN."
    Case "11"
        MsgBox "Entered color is DK_BLUE."
    Case "12"
        MsgBox "Entered color is DK_YELLOW."
    Case "13"
        MsgBox "Entered color is DK_MAGENTA."
    Case "14"
        MsgBox "Entered color is DK_CYAN."
    Case "15"
        MsgBox "Entered color is DK_GRAY."
    Case Else
        MsgBox "Unrecognized entry. Please try again."
        GoTo a:
End Select

Basic_Color = ABC.BasicColor(User_Input)

MsgBox "Long conversion of color is " + CStr(Basic_Color)    ' Display return value
```

## CascadeCharts Method

Related

Related

Related

**Usage** *ApplicationObject.CascadeCharts*

**Description** When you have several ABC chart windows open, you can arrange them in the ABC window using the **CascadeCharts** method. The charts are arranged so the title bar of each one is visible.

**ABC Equivalent** The **CascadeCharts** method is equivalent to choosing Cascade in the ABC Window menu.

## **Related Topics**

### **Language Elements**

[TileCharts Method](#)

### **Description**

[Arranging ABC Charts](#)

### **Object**

[Application Object](#)

**Related**

## CascadeCharts Method Example

This example uses the **CascadeCharts** method of the Application object to cascade all open charts.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True

For I = 1 To 5
    ABC.New
Next I

ABC.CascadeCharts
```

' Start ABC  
' Make ABC visible  
  
' Create a series of new charts  
  
' Cascade all open charts

## CloseAll Method

Related

Related

Related

**Usage** *ApplicationObject.CloseAll*

*ChartsCollection.CloseAll*

**Description** You use the **CloseAll** method to close all charts in the ABC workspace. If changes have been made, the user is not prompted to save changed charts.

**ABC Equivalent** The **CloseAll** method is equivalent to choosing Close All in the File menu, except that the user is not prompted to save changes to changed charts.

## **Related Topics**

### **Language Elements**

[CloseChart Method](#)

[Save Method](#)

### **Description**

[Closing Charts](#)

### **Object**

[Application Object](#)

**Related**

## CloseAll Method Example

This example uses the **CloseAll** method of the Application object to close all charts. Also see the example included with the [Add method of the Charts collection](#).

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
For I = 1 To 5
    ABC.New
Next I
ABC.CloseAll
```

' Start ABC  
' Make ABC visible  
' Create a series of new charts  
' Close all charts

## Help Method

Related

Related

Related

### Usage

*ApplicationObject*.**Help** [*HelpFileName*] {, [*ContextID*] | [*HelpContext*]}

The *HelpFileName* element, an optional string, is the name of a Windows help file.

The *ContextID* element, an optional integer, is a context ID.

The *HelpContext* element, an optional string, specifies the help context to display. In Help, you see the *HelpContext* elements in the top list box when you click the Search button.

### Description

The **Help** method lets you run a help file. The first element specifies the help file to run. If you omit the first element, the help file shipped with ABC runs. The second element is either a context ID (an integer) or a help context (a string) to call a particular topic in the help file. If you omit the element, the Contents of the help file appears.

### ABC Equivalent

If you use the **Help** method to run the help file that ships with ABC, the method is equivalent to pressing **F1** in the proper context. If you are running a help file that you created, there is no ABC equivalent.

## Related Topics

### Description

[Displaying Help](#)

### Object

[Application Object](#)

**Related**

## Help Method Example

This example uses the **Help** method of the Application object to run the ABC help at a specific help topic.

```
Dim ABC As Object
```

```
Set ABC = CreateObject("ABCFlow.application")
```

```
' Start ABC
```

```
ABC.Help , 57345
```

```
' Run ABC Help at Glossary topic
```

## HidePercentGauge Method

Related

Related

Related

**Usage** *ApplicationObject.HidePercentGauge*

**Description** The **HidePercentGauge** method lets you close the Percent Gauge dialog box you created using the **PercentGauge** method.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Hint Method](#)

[MsgBox Method](#)

[PercentGauge Method](#)

[PercentGaugeCancelled Method](#)

[PercentGaugeValue Property](#)

[StatusBar Property](#)

### Description

[Providing Feedback](#)

### Object

[Application Object](#)

**Related**

## HidePercentGauge Method Example

This example uses the **HidePercentGauge** method of the Application object to remove a Percent Gauge dialog box.

```
Dim ABC As Object
Dim ABCGauge As Single

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible

ABC.PercentGauge                                   ' Create gauge

ABC.HidePercentGauge                               ' Remove gauge

MsgBox "Percentage Gauge Hidden"
```

## New Method

Related

Related

Related

**Usage** *ApplicationObject.New*

**Description** You use the **New** method to create a new chart with default attributes. This opens a new chart window.

**Data Type** Object

**Value** The chart object

**ABC Equivalent** The **New** method is equivalent to choosing the New command in the File menu.

## **Related Topics**

### **Language Elements**

[NewFromTemplate Method](#)

### **Description**

[Creating New Charts](#)

### **Object**

[Application Object](#)

**Related**

## New Method Example

This example uses the **New** method of the Application object to create a new chart.

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New ' Create a new chart
```

## NewFromTemplate Method

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject</i> . <b>NewFromTemplate</b> ( <i>TemplateName</i> ) The <i>TemplateName</i> element is the path and name of the template to use to create the chart.
<b>Description</b>	You use the <b>NewFromTemplate</b> method to create a new chart based on the specified chart template name. If <i>TemplateName</i> file cannot be loaded for any reason, the returned Chart.Valid is False.
<b>Data Type</b>	Object
<b>Value</b>	The chart that is created
<b>ABC Equivalent</b>	The <b>AddFromTemplate</b> method is equivalent to choosing the New From Template command in the File menu.

## **Related Topics**

### **Language Elements**

[New Method](#)

### **Description**

[Creating New Charts](#)

### **Object**

[Application Object](#)

**Related**

## NewFromTemplate Method Example

This example uses the **NewFromTemplate** method of the Application object to create a new file using a template.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

ABC.NewFromTemplate ("c:\abc\samples\portmono.aft")    ' Open ABC template file
```

## Open Method

Related

Related

Related

Related

### Usage

*ChartsCollection*.**Open** (*PathName* [, *AsReadOnly*])

*ApplicationObject*.**Open** (*PathName* [, *AsReadOnly*])

The *PathName* element is the path and name of the chart to open.

The *AsReadOnly* element opens the chart as read only.

### Description

The **Open** method in the Charts collection and Application object work the same way and have the same effect. You use the **Open** method to open a chart. If the chart is already open, the **Open** method moves the chart to the front. You can optionally specify that the chart is to be opened read only.

You can open the following file types.

- Charts (filenames ending with an AF3, AF2, or ABC extension; files that contain the shapes, lines, and text that comprise your charts)
- Templates (filenames ending with an AFT extension; files that hold object attributes and page layouts used by your charts)

### Data Type

Object. The *AsReadOnly* element is an integer (Boolean)

### Value

The Chart object

### ABC Equivalent

The **Open** method is equivalent to choosing Open in the File menu, choosing the drive and directory that contain the file you want to open, choosing the file you want to open, and choosing OK.

## Related Topics

### Language Elements

[DefaultFilePath Property](#)

### Description

[Opening Charts](#)

### Object

[Application Object](#)

[Charts Collection](#)

## Related

### Open Method Example

This example uses the **Open** method of the Application object to open a file read/write and then open a file read only.

```
Dim ABC As Object
Dim ABC_Read_Only As Single

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                               ' Make ABC visible

ABC_Read_Only = False                            ' Set parameter to read/write access

ABC.Open "c:\abc\samples\video.af3", ABC_Read_Only ' Open file read/write

ABC_Read_Only = True                             ' Set parameter to read only access

ABC.Open "c:\abc\samples\orglink.af3", ABC_Read_Only ' Open file read only
```



## Quit Method

Related

Related

Related

**Usage** *ApplicationObject.Quit*

**Description** The **Quit** method closes ABC. It does not prompt the user to save changes to open files. Before you close ABC, you should save the files you want to be saved as described in [Working with Chart Files](#).

**ABC Equivalent** The **Quit** method is equivalent to choosing Exit in the ABC File menu.

## Related Topics

**Language Elements**

[CloseAll Method](#)

**Description**

[Closing ABC](#)

**Object**

[Application Object](#)

**Related**

## Quit Method Example

This example uses the **Quit** method of the Application object to close ABC.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible

For I = 1 To 3                                     ' Create a series of new charts
    ABC.New
Next I

ABC.Quit                                           ' Close ABC
```

## MakeRGB Method

Related

Related

Related

### Usage

*ApplicationObject*.**MakeRGB** (*Red, Green, Blue*)

The *Red*, *Green*, and *Blue* elements are integers that define the RGB components of color.

### Description

The **MakeRGB** method lets you set colors from a palette of over sixteen million colors. You specify the color as quantities of red, green, and blue, with each color a number from 0 (no color) through 255 (solid color).

### Data Type

Long

### Value

Returns the decimal equivalent of a six-digit, hexadecimal value. The following table shows some of the values of the red, green, and blue components and their equivalent in decimal and hexadecimal.

Color	MakeRGB	Decimal	Hex
White	(255,255,255)	16777215	FFFFFF
Black	(0,0,0)	0	0
Red	(255,0,0)	255	FF
Green	(0,255,0)	65280	FF00
Blue	(0,0,255)	16711680	FF0000
Yellow	(255,255,0)	65535	FFFF
Magenta	(255,0,255)	16711935	FF00FF
Cyan	(0,255,255)	16776960	FFFF00
Gray	(192,192,192)	12632256	C0C0C0
Dark Red	(127,0,0)	127	7F
Dark Green	(0,127,0)	32512	7F00
Dark Blue	(0,0,127)	8323072	7F0000
Dark Yellow	(127,127,0)	326397	7F7F
Dark Magenta	(127,0,127)	8323199	7F007F
Dark Cyan	(0,127,127)	8355584	7F7F00
Dark Gray	(127,127,127)	8355711	7F7F7F

ABC Equivalent None

## Related Topics

### Language Elements

[BasicColor Method](#)

### Description

[RGB Values](#)

[Text Color](#)

### Object

[Application Object](#)

**Related**

## MakeRGB Method Example

This example uses the **MakeRGB** method of the Application object to find a color value.

```
Dim ABC As Object
Dim Red_Green_Blue As Long

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC

Red_Green_Blue = ABC.MakeRGB(255, 255, 255)           ' Find color value
MsgBox CStr(Red_Green_Blue)
```

## PercentGauge Method

Related

Related

Related

### Usage

*ApplicationObject*.**PercentGauge** [*TitleBar*] [, *TextLine1*] [, *TextLine2*]

The optional *TitleBar* element is the name that goes in the title bar.

The optional *TextLine1* element is the first line of text above the gauge.

The optional *TextLine2* element is the second line of text above the title bar.

### Description

The **PercentGauge** method lets you create a percent gauge, with its value set to 0.

**ABC Equivalent** None

## Related Topics

### Language Elements

[HidePercentGauge Method](#)

[Hint Method](#)

[Hourglass Property](#)

[MsgBox Method](#)

[PercentGaugeCancelled Method](#)

[PercentGaugeValue Property](#)

[StatusBar Property](#)

### Description

[Providing Feedback](#)

### Object

[Application Object](#)

## Related

### PercentGauge Method, PercentGaugeValue Property Example

This example uses the **PercentGauge** method and **PercentGaugeValue** property of the Application object to create and increment a gauge.

```
Dim ABC As Object
Dim ABCGauge As Single

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

ABC.PercentGauge                                       ' Make gauge visible

For I = 1 To 100                                       ' Incrementally increase gauge value
    ABC.PercentGaugeValue = ABC.PercentGaugeValue + 1
Next I
```

## PercentGaugeCancelled Method

Related

Related

Related

**Usage** *ApplicationObject.PercentGaugeCancelled*

**Description** The **PercentGaugeCancelled** method lets you determine whether the user has clicked the Cancel button in the Percent Gauge dialog box you created using the PercentGauge method.

**Data Type** Integer (Boolean)

**Value** True means the user clicked the Cancel button; False means the user did not click the Cancel button.

**ABC Equivalent** None

## Related Topics

### Language Elements

[HidePercentGauge Method](#)

[Hint Method](#)

[MsgBox Method](#)

[PercentGauge Method](#)

[PercentGaugeValue Property](#)

[StatusBar Property](#)

### Description

[Providing Feedback](#)

### Object

[Application Object](#)



## TileCharts Method

Related

Related

Related

**Usage** *ApplicationObject.TileCharts*

**Description** When you have several ABC chart windows open, you can arrange them in the ABC window using the **TileCharts** method. The charts are arranged so that a portion of each is visible.

**ABC Equivalent** The **TileCharts** method is equivalent to choosing Tile in the ABC Window menu.

## Related Topics

### Language Elements

[CascadeCharts Method](#)

### Description

[Arranging ABC Charts](#)

### Object

[Application Object](#)



## Undo Method

Related

Related

Related

**Usage** *ApplicationObject.Undo*

**Description** You use the **Undo** method to undo the last ABC action. You can find out if there is anything to undo using the **UndoAvailable** property.

**ABC Equivalent** The **Undo** method is equivalent to choosing Undo in the ABC Edit menu.

## Related Topics

### Language Elements

[UndoAvailable Property](#)

### Description

[Undoing Actions](#)

### Object

[Application Object](#)

## Related

# Undo Method Example

This example uses the **Undo** method of the Application object to undo a user action.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Msg1 As String, Msg2 As String, Msg3 As String

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                           ' Get the active chart

Set NewShape = Chart.DrawShape("Operation")            ' Draw a shape
NewShape.Color = ABC.MakeRGB(0, 0, 255)               ' Make the shape blue
Msg1 = "A blue shape has just been drawn. "
Msg2 = "Please move this message out of the way and move or delete the shape. "
Msg3 = "Then click OK in this message. The code will undo whatever you just did."
MsgBox Msg1 + Msg2 + Msg3

If ABC.UndoAvailable Then                              ' If undo is available
    ABC.Undo                                           ' Undo the last action
Else
    MsgBox "There is nothing to undo!"
End If
```

## MsgBox Method

Related

Related

Related

### Usage

*ApplicationObject*.**MsgBox** *MessageText* [,*BoxType*] [,*BoxTitle*]

The *MessageText* element is the message that goes in the dialog box.

The optional *BoxType* element defines the type of dialog box. If you omit this element, the value is 0.

The optional *BoxTitle* element sets the title bar text of the dialog box. If you omit this element, the title of the dialog box is "Micrografx ABC FlowCharter."

### Description

The **MsgBox** method lets you post a dialog box. The method is similar to the MsgBox function used in the Visual Basic programming language

### Data Type

Integer

### Value

The following table shows the value returned according to the button that the user selected. **Note:** In Visual Basic, these values have constants associated with them, such as IDOK. Those constants are not available for ABC OLE Automation.

Button Selected	Value
-----------------	-------

OK	1
Cancel	2
Abort	3
Retry	4
Ignore	5
Yes	6
No	7

The following table shows the values available for the *BoxType* element, which is optional. The value of the second element can be the sum of values from the table.

**Note:** In Visual Basic, these values have constants associated with them, such as MB\_OK. Those constants are not available for ABC OLE Automation.

Value	Effect
-------	--------

0	Display OK button only
1	Display OK and Cancel buttons
2	Display Abort, Retry, and Ignore buttons
3	Display Yes, No, and Cancel buttons
4	Display Yes and No buttons
5	Display Retry and Cancel buttons
16	Display stop icon

32 Display question mark icon [Related](#)

48 Display exclamation point icon [Related](#)

64 Display information icon [Related](#)

0 First button is the default

256 Second button is the default

512 Third button is the default

0 The dialog box is application modal, so ABC is suspended until the user responds to the dialog box

4096 The dialog box is system modal, so all applications are suspended until the user responds to the dialog box

**ABC Equivalent** None

## Related Topics

### Description

[Providing Feedback](#)

### Object

[Application Object](#)

## RegisterEvent Method

Related

Related

Related

### Usage

*ApplicationObject*.**RegisterEvent** *VBXName.VBX, IdString, EventName* [, *ChartType*]

The *VBXName.VBX* element identifies the ABC OLE Automation control to which the registered events apply. Unless you have changed the ABC OLE Automation control's **Name** property from its default setting, *VBXName* is ABC1.

The *IdString* element identifies the Visual Basic form on which the ABC OLE Automation control is located. *IdString* identifies the Visual Basic form on which the ABC control is located. It is normally the **Caption** property setting of the form.

The *EventName* element is the name of the event being registered. This name must be enclosed in quotes.

The *ChartType* element, which is optional, lets you register the event for only a particular type of chart. You set a chart's type with the **Type** method of the Chart object. If you omit the *ChartType*, the registered events apply to all charts.

### Description

The **RegisterEvent** method lets you register an event procedure. If you do not register an event, ABC OLE Automation does not respond when the user of your program performs the event.

### Data Type

Integer (Boolean)

### Value

True means the event was successfully registered; False means it was not.

### ABC Equivalent

None

## Related Topics

### Language Elements

[Type Property \(Chart Object\)](#)

[TypeRequiresEXE Property](#)

[TypeUsesEXE Property](#)

[UnRegisterEvent Method](#)

### Description

[Registering Event Procedures](#)

### Object

[Application Object](#)

## Related

# RegisterEvent, UnRegisterEvent Method Example

This example uses the **RegisterEvent** method and **UnRegisterEvent** method of the Application object to register an event. The code must be in three different places in Visual Basic.

The first section of code goes in Form\_Load or in a Command button.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Msg1 As String, Msg2 As String

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

' Begin accepting user actions
ABC.RegisterEvent ABC1.VBX, Form1.Caption, "ObjectMovedNOTIFY"
ABC.RegisterEvent ABC1.VBX, Form1.Caption, "ObjectLineAttachNOTIFY"

Msg1 = "After you click OK, two shapes will appear. Please move either shape once. "
Msg2 = "This application will know when you have moved one, and you will see it change."
MsgBox Msg1 + Msg2
Set Shape1 = Chart.DrawShape("Operation")             ' Draw shapes
Set Shape2 = Chart.DrawShape("Decision")
Shape1.Text = "Move me!"                               ' Place text in the shapes
Shape2.Text = "No! Move me!"
Shape2.Shape.FitShapeToText                           ' Resize the shape so its text fits

End Sub
```

The second section of code goes in ABC1\_ObjectMovedNOTIFY.

```
Dim ABC As Object
Dim ABCObj As Object

Set ABC = CreateObject("ABCFlow.application")         ' Start ABC
Set ABCObj = ABC1.Object                              ' Get the object that was moved

ABCObj.Color = ABC.MakeRGB(255, 0, 0)                 ' Draw a shape
' Place text in the shape
ABCObj.Text = "Eeek! I've been moved! Please draw a line from me to the other shape."
ABCObj.Shape.FitShapeToText                           ' Resize the shape so its text fits
```

The third section of code goes in ABC1\_ObjectLineAttachedNOTIFY.

```
Dim ABC As Object
Set ABC = CreateObject("ABCFlow.application")         ' Start ABC

MsgBox "This application will now stop receiving ABC events."

' Stop accepting user actions
ABC.UnRegisterEvent ABC1.VBX, "ObjectMovedNOTIFY"
ABC.UnRegisterEvent ABC1.VBX, "ObjectLineAttachNOTIFY"
```

## UnRegisterEvent Method

Related

Related

Related

### Usage

*ApplicationObject*.**UnRegisterEvent** *VBXName.VBX, EventName* [, *ChartType*]

The *VBXName.VBX* element identifies the ABC OLE Automation control to which the registered events apply. Unless you have changed the ABC OLE Automation control's **Name** property from its default setting, *VBXName* is ABC1.

The *EventName* element is the name of the event being unregistered. This name must be enclosed in quotes.

The *ChartType* element, which is optional, lets you unregister the event for only a particular type of chart. You set a chart's type with the **Type** method of the Chart object. If you omit the *ChartType*, the unregister applies to all charts.

### Description

The **UnRegisterEvent** method lets you unregister an event procedure. If you do not unregister an event, ABC OLE Automation continues to respond when the user of your program performs the event.

### Data Type

Integer (Boolean)

### Value

True means the event was successfully unregistered; False means it was not.

### ABC Equivalent

None

## Related Topics

### Language Elements

[RegisterEvent Method](#)

[Type Property \(Chart Object\)](#)

[TypeRequiresEXE Property](#)

[TypeUsesEXE Property](#)

### Description

[Registering Event Procedures](#)

### Object

[Application Object](#)

## CreateAddOn Method

Related

Related

Related

### Usage

*ApplicationObject*.**CreateAddOn** *Position*, *HintName*, *ProgramFileName* [, *IconFileName*]

The *Position* element specifies the position of the button. It can be 1 through 10, or you can use -1 for the first available position. **Note:** In ABC Viewer, you can only use the numbers 3 through 10 because numbers 1 and 2 are for the DataAnalyzer button and SnapGraphics button, which are not in ABC Viewer.

The *HintName* element is the name of the button. The text you enter is used in bubble help and in the hint line.

The *ProgramFileName* element is the name of the program to run, including the fully qualified path.

The *IconFileName* element, which is optional, is the location of the icon you want to use for the button, including the fully qualified path. The button is 25 X 25 pixels or 17 X 17 pixels (depending on the version of ABC you are using), so standard icons (which are 32 X 32 pixels) are shrunk to fit. Alternatively, you can provide a BMP file to use as the button. If the BMP file is smaller than required, the bitmap is centered. If you do not provide this element, the icon in the program to run is used.

### Description

The **CreateAddOn** method of the Menu collection lets you add buttons to the toolbox.

### Data Type

Integer (Boolean)

### Value

True means the button was created successfully; False means it was not.

### ABC Equivalent

None

## **Related Topics**

### **Language Elements**

[AddMenu Method](#)

[RemoveAddOn Method](#)

### **Description**

[Adding Buttons](#)

### **Object**

[Application Object](#)



## RemoveAddOn Method

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.RemoveAddOn</i> { <i>Position</i>   <i>ProgramFileName</i> }
	The <i>Position</i> element specifies the position of the button. It can be 1 through 6. The <i>ProgramFileName</i> element is the name of the executable file, including the fully qualified path.
<b>Description</b>	The <b>RemoveAddOn</b> method lets you remove a button that you have added to the ABC toolbox.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the button was removed successfully; False means it was not.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[CreateAddOn Method](#)

[RemoveMenu Method](#)

### Description

[Adding Buttons](#)

### Object

[Application Object](#)

## Restore Method (Application Object)

Related

Related

Related

**Usage** *ApplicationObject.Restore*

**Description** The **Restore** method of the Application object lets you change the ABC window to its previous size.

**ABC Equivalent** The **Restore** method is equivalent to clicking the ABC restore arrow in the upper right of the window.

## Related Topics

### Language Elements

[Maximize Method \(Application Object\)](#)

[Minimize Method \(Application Object\)](#)

[Restore Method \(Chart Object\)](#)

### Description

[Minimizing, Maximizing, and Restoring a Window](#)

### Object

[Application Object](#)

## Minimize Method (Application Object)

Related

Related

Related

### Usage

*ApplicationObject.Minimize*

### Description

The **Minimize** method of the Application object lets you change the ABC window to an icon.

### ABC Equivalent

The **Minimize** method is equivalent to clicking the ABC minimize arrow in the upper right of the window.

## Related Topics

### Language Elements

[Maximize Method \(Application Object\)](#)

[Minimize Method \(Chart Object\)](#)

[Restore Method \(Application Object\)](#)

### Description

[Minimizing, Maximizing, and Restoring a Window](#)

### Object

[Application Object](#)

## Maximize Method (Application Object)

Related

Related

Related

### Usage

*ApplicationObject.Maximize*

### Description

The **Maximize** method of the Application object lets you change the ABC window to its maximum size.

### ABC Equivalent

The **Maximize** method is equivalent to clicking the ABC maximize arrow in the upper right of the window.

## **Related Topics**

### **Language Elements**

[Maximize Method \(Chart Object\)](#)

[Minimize Method \(Application Object\)](#)

[Restore Method \(Application Object\)](#)

### **Description**

[Minimizing, Maximizing, and Restoring a Window](#)

### **Object**

[Application Object](#)

## Related

### Maximize, Minimize, Restore Method (Application Object) Example

This example uses the **Maximize** method, **Minimize** method, and **Restore** method of the Application object to minimize, restore, and maximize the ABC window.

```
Dim ABC As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                               ' Make ABC visible

MsgBox "Click OK to minimize ABC FlowCharter."
ABC.Minimize                                     ' Minimize ABC

MsgBox "Click OK to maximize ABC FlowCharter."
ABC.Maximize                                    ' Maximize ABC

MsgBox "Click OK to restore ABC FlowCharter to normal size."
ABC.Restore                                     ' Restore ABC
```

## Hourglass Property

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject.Hourglass</i> = {True   False}
<b>Description</b>	The <b>Hourglass</b> property lets you change the pointer to a wait cursor or back to the ABC pointer. The <b>Hourglass</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes the pointer a wait cursor; False makes the cursor the ABC pointer.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Hint Method](#)

[PercentGauge Method](#)

[StatusBar Property](#)

### Description

[Providing Feedback](#)

### Object

[Application Object](#)



## AddMenu Method

Related

Related

Related

### Usage

*ApplicationObject*.**AddMenu** (*MenuName*, *VBXName.VBX*, *ProgramName* [, *ChartType*])

The *MenuName* element is the title of the menu.

The *VBXName.VBX* element identifies the VBX used to send notification events to when the menu is used. Normally you use ABC1.VBX, which registers menus for the **AppMenuSUBCLASS** event.

The *ProgramName* element is the name of the program adding the menu to ABC. The easiest way to identify the program is using Form1.Caption.

The *ChartType* element, which is optional, lets you specify a chart type for the menu. A chart type is a hidden string field up to eight characters in length indicating the chart type. This field is never used within ABC, but it is useful within an ABC events VBX. For example, if two OLE Automation programs are running, you could change the fourth element to avoid conflicts.

### Description

The **AddMenu** method lets you add a menu to ABC. The menu is added to ABC at the left of the Window menu, so you set the order of the menus by the order in which you create them. When the VBX shuts down (when the program ends), the menu is removed from ABC. (See [Handling ABC Events](#) for more information on event handling.)

### Data Type

Object

### Value

The Menu object

### ABC Equivalent

None

## Related Topics

### Language Elements

[AppendItem Method](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[InsertItem Method](#)

[RemoveMenu Method](#)

[Text Property \(Menu Collection\)](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Adding Menus](#)

### Object

[Application Object](#)

**Related**

## AddMenu Method, AppendItem Method Example

This example uses the **AddMenu** method of the Application object and the **AppendItem** method of the Menu collection to create a menu and add two items to it.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

Set Menu = ABC.AddMenu("Test", ABC1.VBX, Form1.Caption) ' Add a new menu item

Menu.AppendItem ("First Item")                         ' Append items to the new menu
Menu.AppendItem ("Second Item")
```

## RemoveMenu Method

Related

Related

Related

**Usage** *ApplicationObject.RemoveMenu MenuName*  
The *MenuName* element is the title of the menu.

**Description** The **RemoveMenu** method lets you remove a menu you have added. You cannot remove the menus that ABC starts with.

**ABC Equivalent** None

## Related Topics

### Language Elements

[AddMenu Method](#)

[AppendItem Method](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[InsertItem Method](#)

[Text Property \(Menu Collection\)](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Adding Menus](#)

### Object

[Application Object](#)



## Hint Method

Related

Related

Related

### Usage

*ApplicationObject.Hint HintText*

The *HintText* element is the text of the message you are placing in the hint line.

### Description

The **Hint** method lets you set a temporary status bar message. It stays in the hint line until the cursor moves over another item in ABC that causes the hint line to change. To set a permanent message in the hint line, use the **StatusBar** property.

**ABC Equivalent** None

## Related Topics

### Language Elements

[AddMenu Method](#)

[CreateAddOn Method](#)

[MsgBox Method](#)

[PercentGauge Method](#)

[StatusBar Property](#)

### Description

[Providing Feedback](#)

### Object

[Application Object](#)

## Related

### Hint Method Example

This example uses the **Hint** method of the Application object to display a hint message.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

Set NewShape = Chart.DrawShape("Operation")             ' Draw a shape
NewShape.Text = "Look at the hint line!"               ' Place text in the shape
NewShape.Shape.ShadowStyle = 3                         ' Give the shape a shadow
' Display a hint line
ABC.Hint "Poltergeists are the principal form of supernatural manifestation."
```

## ChartTypeShutdown Method

Related

Related

Related

<b>Usage</b>	<i>ApplicationObject</i> . <b>ChartTypeShutdown</b> <i>ChartType</i> , <i>ApplicationName</i> The <i>ChartType</i> element is the type of chart that you want to close. The <i>ApplicationName</i> element is the name of the application.
<b>Description</b>	The <b>ChartTypeShutdown</b> method lets you have an external program (EXE) alert ABC OLE Automation that it is shutting down for some reason, usually because a RegisterEvent call failed. You usually call the <b>ChartTypeShutdown</b> method during the loading of an AddOn menu if a RegisterEvent call failed. You can also call it from the Form.QueryUnload event indicating that the external program with SUBCLASSing behavior is shutting down. If you set either <b>TypeRequiresEXE</b> or <b>TypeUsesEXE</b> to True in a program, then you also must ensure that you close all charts of that <b>Type</b> when your program closes. You use the <b>ChartTypeShutdown</b> method of the Application object to close the charts.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[CreateAddOn Method](#)

[RegisterEvent Method](#)

[TypeRequiresEXE Property](#)

[TypeUsesEXE Property](#)

### Description

[Linking EXEs to Charts](#)

### Object

[Application Object](#)

## Related

# ChartTypeShutdown Method Example

This example uses the **ChartTypeShutdown** method of the Application object in two places. When the form loads, **ChartTypeShutdown** is activated if the Events being registered do not register (if an error occurs). Then the **ChartTypeShutdown** method completely unloads all events. When the form unloads, the **ChartTypeShutdown** method is used to completely unregister all events. For this to work, the user must have a form already created that contains this code.

The following code needs to be in the general declarations section of the Form.

```
Const APPNAME = "Sample Application"           ' Sets the APPNAME constant
Const CHARTTYPE = "TEST"                     ' Sets the CHARTTYPE constant
Dim ABC As Object
```

The following code needs to be in the [Form\_Load] routine.

```
Form1.Caption = "Sample Application"           ' Set title bar of sample application

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True                            ' Make ABC visible

ABC.Caption = "Sample application is now running." ' Change ABC title bar

ItFailed = False                             ' Set ItFailed variable to False

' Register Events with ABC FlowCharter 4.0
If (Not ABC.RegisterEvent(ABC1.VBX, APPNAME, "ChartActivateNOTIFY")) Then ItFailed = True
If (Not ABC.RegisterEvent(ABC1.VBX, APPNAME, "ObjectMovedNOTIFY", CHARTTYPE)) Then ItFailed = True
If (Not ABC.RegisterEvent(ABC1.VBX, APPNAME, "ObjectMoveSUBCLASS", CHARTTYPE)) Then ItFailed = True

' Check to see if ItFailed was set to True after registering the Events above
If (ItFailed) Then
    ABC.ChartTypeShutdown CHARTTYPE, APPNAME ' Unregister all registered Events with ABC
FlowCharter 4.0
    MsgBox "Could not register events. Closing application."
    ABC.Caption = "" ' Change ABC title bar
    End ' Close application
End If
```

The following code needs to in the [Form\_QueryUnload] routine.

```
ABC.ChartTypeShutdown CHARTTYPE, APPNAME ' Unregister all registered Events with ABC
FlowCharter 4.0

MsgBox "Unregistering events. Closing application."
ABC.Caption = "" ' Change ABC title bar
End ' Close application
```

## Chart Object

**Description** The Chart object is below the Chart collection. You can have multiple Chart objects. Each Chart object is restricted to a single PageLayout and MasterItems object, but can have multiple FieldTemplate and Object objects.

### Properties

---

[Application](#)  
[ClipboardFormatAvailable](#)  
[CurrentLineRouting](#)  
[CurrentShape](#)  
[CurrentShapePalette](#)  
[DrawDirection](#)  
[DrawPositionX](#)  
[DrawPositionY](#)  
[DrawSpacingX](#)  
[DrawSpacingY](#)  
[FieldFont](#)  
[FieldNamesHidden](#)  
[FieldPlacement](#)  
[FieldsDaysPerWeek](#)  
[FieldsHoursPerDay](#)  
[FieldsOpaque](#)  
[FieldTemplates](#)  
[FullName](#)  
[GuidelinesOn](#)  
[HasDiskFile](#)  
[LaunchIndicator](#)  
[LaunchShadow](#)  
[LineCrossoverSize](#)  
[LineCrossoverStyle](#)  
[LinkIndicator](#)  
[LinkShadow](#)  
[MasterItems](#)  
[Name](#)  
[NextNumber](#)  
[NextShapeHeight](#)  
[NextShapeWidth](#)  
[NoRepaint](#)  
[NoteIndicator](#)  
[NoteShadow](#)  
[NumberFont](#)  
[Objects](#)  
[PageCount](#)  
[PageLayout](#)  
[Parent](#)  
[Protected](#)  
[ReadOnly](#)  
[Saved](#)  
[ScrollLeft](#)  
[ScrollTop](#)  
[SelectedLineCount](#)  
[SelectedObjectCount](#)  
[SelectedOtherCount](#)  
[SelectedShapeCount](#)  
[ShowLegend](#)

### Methods

---

[Activate](#)  
[AddHorizontalGuideline](#)  
[AddVerticalGuideline](#)  
[CancelFullScreen](#)  
[Clear](#)  
[ClearGuidelines](#)  
[CloseChart](#)  
[Copy](#)  
[Cut](#)  
[DeselectAll](#)  
[DrawFreeLine](#)  
[DrawLine](#)  
[DrawLineToOneObject](#)  
[DrawShape](#)  
[DrawTextBlock](#)  
[Duplicate](#)  
[FullScreen](#)  
[GroupAndLink](#)  
[InsertObjectFromFile](#)  
[Minimize](#)  
[Maximize](#)  
[Paste](#)  
[PasteLink](#)  
[PasteSpecial](#)  
[PrintOut](#)  
[PrintSelected](#)  
[Repaint](#)  
[Restore](#)  
[RevertToSaved](#)  
[Save](#)  
[ScrollPage](#)  
[ScrollPosition](#)  
[Select](#)  
[SelectShapeType](#)  
[SendMail](#)  
[SetDefaults](#)  
[SetProtection](#)  
[Spelling](#)  
[ToBack](#)  
[ToFront](#)  
[UpdateFields](#)

ShowNodesOnLines

Type

TypeRequiresEXE

TypeUsesEXE

Units

Valid

View

WindowHandle

ZoomPercentage

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

[VBX Event Variables](#)

## ClipboardFormatAvailable Property

Related

Related

Related

### Usage

*ChartObject.ClipboardFormatAvailable (Format)*

### Description

The **ClipboardFormatAvailable** property lets you find if a format is available to be pasted from the Windows Clipboard. The **ClipboardFormatAvailable** property is read only.

### Data Type

Integer (Boolean)

### Value

True means the format is available; False means the format is not available. The values for the formats are in the following table.

<b>Format</b>	<b>Description</b>
0	ABC Native
1	OLE Client Embed
2	ABC Rich Text
3	Rich Text Format (RTF)
4	Unformatted Text
5	Metafile
6	Device-Independent Bitmap
7	Bitmap
8	OLE Client Link

### ABC Equivalent

The **ClipboardFormatAvailable** property is equivalent to choosing Paste Special in the ABC Edit menu and checking if a format is available.

## **Related Topics**

### **Language Elements**

[Copy Method](#)

[Cut Method](#)

[Paste Method](#)

[PasteSpecial Method](#)

### **Description**

[Using Special Clipboard Formats](#)

### **Object**

[Chart Object](#)

## CurrentLineRouting Property

Related

Related

Related

- Usage** *ChartObject.CurrentLineRouting = LineRoutingValue*
- Description** You use the **CurrentLineRouting** property to find or set the type of routing for new lines. The **CurrentLineRouting** property is read/write.
- Data Type** Integer
- Value** The following table describes the values for the **CurrentLineRouting** property.
- | Value | Type of Line       |
|-------|--------------------|
| 0     | Direct             |
| 1     | Right angle        |
| 2     | Curved             |
| 3     | Organization chart |
| 4     | Cause-and-effect   |
- ABC Equivalent** The **CurrentLineRouting** property is equivalent to clicking the Line tool and then clicking one of the line routing buttons with no lines selected. You cannot change the type of routing for lines that have already been drawn

## Related Topics

### Language Elements

[Type Property \(Line Object\)](#)

### Description

[Setting Line Routing](#)

### Object

[Chart Object](#)

## CurrentShape Property

Related

Related

Related

### Usage

*ChartObject.CurrentShape = Name*

### Description

The **CurrentShape** property lets you find or set the current shape so that it is the next shape drawn when you draw a shape. When you are setting the value, you can define it loosely. For example, setting its value to "dec" chooses "Decision." The **CurrentShape** property is read/write.

### Data Type

String

### Value

The name of the next shape to be drawn

### ABC Equivalent

The **CurrentShape** property is equivalent to clicking the shape you want in the Shape palette.

## Related Topics

### Language Elements

[CurrentShape Property](#)

[DrawShape Method](#)

### Description

[Choosing a Shape in the Palette](#)

### Object

[Chart Object](#)

## Related

### CurrentShape Property and DrawShape Method Example

This example uses the **CurrentShape** property and **DrawShape** method of the Chart object to set the type of shape to be created.

```
Dim ABC As Object, Chart As Object, Obj1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Chart.CurrentShape = "Decision"
Set Obj1 = Chart.DrawShape()
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Set shape to Decision diamond  
' Draw Decision shape

## CurrentShapePalette Property

Related

Related

Related

### Usage

*ChartObject.CurrentShapePalette = ShapePaletteName*

### Description

You use the **CurrentShapePalette** property to open a Shape Palette or determine the name of the current Shape Palette. The name of the Shape Palette appears in the title bar of the palette and is not related to the filename of the palette. The **CurrentShapePalette** property is read/write.

### Data Type

String

### Value

The name of the current shape palette

### ABC Equivalent

The **CurrentShapePalette** property is equivalent to choosing Open in the Shape Palette menu and choosing the palette you want.

## Related Topics

### Language Elements

[CurrentShape Property](#)

[DrawShape Method](#)

### Description

[Using the Shape Palette](#)

### Object

[Chart Object](#)

**Related**

## CurrentShapePalette Property Example

This example uses the **CurrentShapePalette** property of the Chart object to set the current shape palette.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Chart.CurrentShapePalette = "Net - PC"
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
' Change current shape palette

## DrawDirection Property

Related

Related

Related

### Usage

*ChartObject.DrawDirection = Direction*

### Description

The **DrawDirection** property lets you find or set the direction for placing new shapes. The **DrawDirection** property is read/write.

### Data Type

Integer

### Value

The **DrawDirection** property uses the values shown in the following table.

Value	Description
0	North
1	East
2	South
3	West
10	Stacked

**ABC Equivalent** None

## Related Topics

### Language Elements

[DrawShape Method](#)

[StretchType Property](#)

### Description

[Drawing Shapes](#)

### Object

[Chart Object](#)

## DrawPositionX Property

Related

Related

Related

### Usage

*ChartObject.DrawPositionX = HorizontalDistance*

### Description

The **DrawPositionX** property lets you find or set the horizontal drawing position where you want to place the next object, text, or line. The position you specify is used for the next object drawn, or the next object pasted or pasted special (if those methods do not specify a different position). You set the units used to measure the distance using the

**Units** property. The **DrawPositionX** property is read/write.

### Data Type

Double

### Value

The horizontal location for the next drawing position

### ABC Equivalent

None

## Related Topics

### Language Elements

[DrawPositionY Property](#)

[DrawSpacingX Property](#)

[DrawSpacingY Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Setting the Current Drawing Position](#)

[Creating Text Blocks](#)

### Object

[Chart Object](#)

## Related

# DrawPositionX Property, DrawPositionY Property, DrawSpacingX Property, and DrawSpacingY Property Example

This example uses the **DrawPositionX** property, **DrawPositionY** property, **DrawSpacingX** property, and **DrawSpacingY** property of the Chart object to set the position and spacing for drawing shapes.

```
Dim ABC As Object, Chart As Object
Dim Shapes
```

```
Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart
```

```
' Start ABC
' Make ABC visible
' Get the active chart
```

```
Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Chart.DrawSpacingX = 1.5
Chart.DrawSpacingY = w.5
```

```
' Set X coordinate for drawing
' Set Y coordinate for drawing
' Set X coordinate spacing
' Set Y coordinate spacing
```

```
For Shapes = 1 To 4
    Chart.DrawShape ("Storage")
Next Shapes
```

```
' Draw shapes
```

## DrawPositionY Property

Related

Related

Related

### Usage

*ChartObject.DrawPositionY = VerticalDistance*

### Description

The **DrawPositionY** property lets you find or set the vertical drawing position where you want to place the next object, text, or line. The position you specify is used for the next object drawn, or the next object pasted or pasted special (if those methods do not specify a different position). You set the units used to measure the distance using the

**Units** property. The **DrawPositionY** property is read/write.

### Data Type

Double

### Value

The vertical location for the next drawing position

### ABC Equivalent

None

## Related Topics

### Language Elements

[DrawPositionX Property](#)

[DrawSpacingX Property](#)

[DrawSpacingY Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Setting the Current Drawing Position](#)

[Creating Text Blocks](#)

### Object

[Chart Object](#)

## DrawSpacingX Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.DrawSpacingX = Spacing</i>
<b>Description</b>	The <b>DrawSpacingX</b> property lets you find or set the horizontal spacing for the next shape placed. The <b>DrawSpacingX</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The horizontal spacing for the next shape placed
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[DrawPositionX Property](#)

[DrawPositionY Property](#)

[DrawSpacingY Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Drawing Shapes](#)

### Object

[Chart Object](#)

## DrawSpacingY Property

Related

Related

Related

Related

**Usage** *ChartObject.DrawSpacingY = Spacing*

**Description** The **DrawSpacingY** property lets you find or set the vertical spacing for the next shape placed. The **DrawSpacingY** property is read/write.

**Data Type** Double

**Value** The vertical spacing for the next shape placed

**ABC Equivalent** None

## Related Topics

### Language Elements

[DrawPositionX Property](#)

[DrawPositionY Property](#)

[DrawSpacingX Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Drawing Shapes](#)

### Object

[Chart Object](#)

## FieldFont Property

Related

Related

Related

### Usage

*ChartObject*.FieldFont

### Description

The **FieldFont** property lets you find or set properties for the Font object for field text in a chart. All the properties of the Font object, such as bold and italic, are available through the **FieldFont** property. The **FieldFont** property is read only, but all the properties from the object it returns are read/write.

### Data Type

Object

### Value

A Font object

### ABC Equivalent

The **FieldFont** property is equivalent to choosing Preferences in the ABC File menu, clicking the Field Display button, and setting the font attributes in the field font area.

## Related Topics

### Language Elements

[Bold Property](#)

[Color Property \(Font Object\)](#)

[FieldNamesHidden Property](#)

[FieldPlacement Property](#)

[FieldsDaysPerWeek Property](#)

[FieldsHoursPerDay Property](#)

[FieldsOpaque Property](#)

[Italic Property](#)

[Name Property \(Font Object\)](#)

[Opaque Property](#)

[Size Property](#)

[Strikethrough Property](#)

[Underline Property](#)

### Description

[Field Options](#)

[Setting Data Field Preferences](#)

### Object

[Chart Object](#)

## Related

# FieldFont Property Example

This example uses the **FieldFont** property of the Chart object to set the style of the font used for data fields.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Field1 As Object, Field2 As Object, FieldFontStyle As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                               ' Position fields below shapes
Set Field1 = Chart.FieldTemplates.Add("Name")          ' Add a field
Field1.Format = 0                                     ' Format field as text
Field1.AccumulationMethod = 0                         ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")        ' Add a field
Field2.Format = 0                                     ' Format field as text
Field2.AccumulationMethod = 0                         ' No accumulation

Set FieldFontStyle = Chart.FieldFont                  ' Set the FieldFont object
FieldFontStyle.Name = "Roman"                         ' Change the font
FieldFontStyle.Italic = True                          ' Make it italic

Obj1.FieldValues.Item("Name").Value = "Joe Smith"    ' Enter field values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"
```

## FieldNamesHidden Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> .FieldNamesHidden = {True   False}
<b>Description</b>	The <b>FieldNamesHidden</b> property lets you find or set whether field names are shown. The <b>FieldNamesHidden</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True hides field names; False shows them.
<b>ABC Equivalent</b>	The <b>FieldNamesHidden</b> property is equivalent to choosing Preferences in the ABC Fields menu and selecting the Hide Field Names option (True) or deselecting the option (False).

## Related Topics

### Language Elements

[FieldFont Property](#)

[FieldPlacement Property](#)

[FieldsDaysPerWeek Property](#)

[FieldsHoursPerDay Property](#)

[FieldsOpaque Property](#)

[Hidden Property](#)

### Description

[Field Options](#)

[Setting Data Field Preferences](#)

### Object

[Chart Object](#)

**Related**

## FieldNamesHidden Property Example

This example uses the **FieldNamesHidden** property of the Chart object to hide data field names.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Field1 As Object, Field2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                               ' Position fields below shapes
Set Field1 = Chart.FieldTemplates.Add("Name")          ' Add a field
Field1.Format = 0                                     ' Format field as text
Field1.AccumulationMethod = 0                         ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")        ' Add a field
Field2.Format = 0                                     ' Format field as text
Field2.AccumulationMethod = 0                         ' No accumulation

Obj1.FieldValues.Item("Name").Value = "Joe Smith"    ' Enter field values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"

Chart.FieldNamesHidden = True                         ' Hide the field labels
```

## FieldPlacement Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.FieldPlacement = Value</i>														
<b>Description</b>	The <b>FieldPlacement</b> property lets you specify the field placement in relation to shapes. The <b>FieldPlacement</b> property is read/write.														
<b>Data Type</b>	Integer														
<b>Value</b>	The values for the field placements are in the following table.														
	<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Left</td></tr><tr><td>1</td><td>Right</td></tr><tr><td>2</td><td>Above</td></tr><tr><td>3</td><td>Below</td></tr><tr><td>4</td><td>Inside Top</td></tr><tr><td>5</td><td>Inside Middle</td></tr></tbody></table>	Value	Description	0	Left	1	Right	2	Above	3	Below	4	Inside Top	5	Inside Middle
Value	Description														
0	Left														
1	Right														
2	Above														
3	Below														
4	Inside Top														
5	Inside Middle														
<b>ABC Equivalent</b>	The <b>FieldPlacement</b> property is equivalent to choosing Preferences in the ABC Fields menu and choosing a placement location in the Field Placement area.														

## Related Topics

### Language Elements

[FieldFont Property](#)

[FieldNamesHidden Property](#)

[FieldsDaysPerWeek Property](#)

[FieldsHoursPerDay Property](#)

[FieldsOpaque Property](#)

### Description

[Field Options](#)

[Setting Data Field Preferences](#)

### Object

[Chart Object](#)

## Related

# FieldPlacement Property Example

This example uses the **FieldPlacement** property of the Chart object to put data fields below shapes.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Field1 As Object, Field2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                               ' Position fields below shapes

Set Field1 = Chart.FieldTemplates.Add("Name")          ' Add a field
Field1.Format = 0                                     ' Format field as text
Field1.AccumulationMethod = 0                         ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")        ' Add a field
Field2.Format = 0                                     ' Format field as text
Field2.AccumulationMethod = 0                         ' No accumulation

Obj1.FieldValues.Item("Name").Value = "Joe Smith"    ' Enter field values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"
```

## FieldTemplates Property

Related

Related

Related

**Usage** *ChartObject.FieldTemplates*

**Description** You use the **FieldTemplates** property to find the FieldTemplates collection. The **FieldTemplates** property is read only, but the properties from the collection it returns are read/write.

**Data Type** Collection object

**Value** The FieldTemplates collection

**ABC Equivalent** None

## Related Topics

### Description

[Adding Data Fields to a Chart](#)

### Object

[Chart Object](#)

## FieldsHoursPerDay Property

Related

Related

Related

### Usage

*ChartObject.FieldsHoursPerDay = HoursPerDay*

### Description

The **FieldsHoursPerDay** property lets you find or set the number of hours in a workday. This value is used when a field is converted between hours and days. The value can range from 1 to 24. For example, the value is used if you change the data field's format from hours to days or you link to a chart that displays data fields in a different format. The **FieldsHoursPerDay** property is read/write.

### Data Type

Integer

### Value

The number of hours in a day for fields

### ABC Equivalent

The **FieldsHoursPerDay** property is equivalent to choosing Preferences in the ABC Fields menu and entering a number in the Hours Per Day area.

## Related Topics

### Language Elements

[FieldFont Property](#)

[FieldNamesHidden Property](#)

[FieldPlacement Property](#)

[FieldsDaysPerWeek Property](#)

[FieldsOpaque Property](#)

### Description

[Field Options](#)

[Setting Data Field Preferences](#)

### Object

[Chart Object](#)

## Related

### FieldsHoursPerDay, FieldsDaysPerWeek Properties Example

This example uses the **FieldsHoursPerDay** property and **FieldsDaysPerWeek** property of the Chart object to find and change the hours per day and days per week.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object
Dim Shape1 As Object, Shape2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Text1 = Chart.DrawTextBlock("Current Field Options") ' Place a text block

Chart.DrawPositionX = 1                               ' Set horizontal position
Chart.DrawPositionY = 2.5                             ' Set vertical position
Set Shape1 = Chart.DrawShape("Operation")             ' Place shapes on the chart
Set Shape2 = Chart.DrawShape("Operation")
Shape1.Text = "Hours per day = " + Chart.FieldsHoursPerDay ' Display hours per day
Shape2.Text = "Days per week = " + Chart.FieldsDaysPerWeek ' Display days per week

MsgBox "Click OK to change the number of " + Chr$(13) + "hours per day and the days per week."
' Chr$(13) is Carriage Return
Chart.FieldsHoursPerDay = 24                          ' Change hours per day
Chart.FieldsDaysPerWeek = 7                          ' Change days per week
Shape1.Text = "Hours per day = " + Chart.FieldsHoursPerDay ' Display hours per day
Shape2.Text = "Days per week = " + Chart.FieldsDaysPerWeek ' Display days per week
```

## FieldsDaysPerWeek Property

Related

Related

Related

### Usage

*ChartObject.FieldsDaysPerWeek = DaysPerWeek*

### Description

The **FieldsDaysPerWeek** property lets you find or set the number of days in a workweek. This value is used when a field is converted between days and weeks. The value can range from 1 to 7. For example, the value is used if you change the data field's format from days to weeks or you link to a chart that displays data fields in a different format. The **FieldsDaysPerWeek** property is read/write.

### Data Type

Integer

### Value

The number of days in a week for fields

### ABC Equivalent

The **FieldsDaysPerWeek** property is equivalent to choosing Preferences in the ABC Fields menu and entering a number in the Days Per Week area.

## Related Topics

### Language Elements

[FieldFont Property](#)

[FieldNamesHidden Property](#)

[FieldPlacement Property](#)

[FieldsHoursPerDay Property](#)

[FieldsOpaque Property](#)

### Description

[Field Options](#)

[Setting Data Field Preferences](#)

### Object

[Chart Object](#)

## FieldsOpaque Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.FieldsOpaque</i> = {True   False}
<b>Description</b>	The <b>FieldsOpaque</b> property lets you find or set whether fields are opaque. The <b>FieldsOpaque</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes the background opaque; False makes the background transparent.
<b>ABC Equivalent</b>	The <b>FieldsOpaque</b> property is equivalent to choosing Preferences in the ABC Fields menu and selecting or deselecting the Opaque Fields option.

## **Related Topics**

### **Language Elements**

[FieldFont Property](#)

[FieldNamesHidden Property](#)

[FieldPlacement Property](#)

[FieldsDaysPerWeek Property](#)

[FieldsHoursPerDay Property](#)

### **Description**

[Field Options](#)

[Setting Data Field Preferences](#)

### **Object**

[Chart Object](#)

**Related**

## FieldsOpaque Property Example

This example uses the **FieldsOpaque** property of the Chart object to make data fields opaque.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object, Obj3 As Object, Obj4 As Object
Dim Field1 As Object, Field2 As Object, A As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Decision")                 ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.FieldPlacement = 3                               ' Position fields below shapes
Set Field1 = Chart.FieldTemplates.Add("Name")          ' Add field
Field1.Format = 0                                     ' Format field as text
Field1.AccumulationMethod = 0                         ' No accumulation
Set Field2 = Chart.FieldTemplates.Add("Phone")        ' Add field
Field2.Format = 0                                     ' Format field as text
Field2.AccumulationMethod = 0                         ' No accumulation

Obj1.FieldValues.Item("Name").Value = "Joe Smith"     ' Set Field Values
Obj1.FieldValues.Item("Phone").Value = "555-1212"
Obj2.FieldValues.Item("Name").Value = "Jane Doe"
Obj2.FieldValues.Item("Phone").Value = "555-1234"

Chart.DrawPositionX = 1                               ' Set draw position
Chart.DrawPositionY = 2
Set Obj3 = Chart.DrawShape("Operation")               ' Draw shapes
Set Obj4 = Chart.DrawShape("Operation")
Obj3.Shape.FillColor = ABC.BLUE                       ' Color the shapes
Obj4.Shape.FillColor = ABC.RED

Chart.SelectShapeType ("Operation")                  ' Select shapes
Chart.ToBack                                          ' Move to back

Chart.FieldsOpaque = True                             ' Make field text opaque
```

## FullName Property (Chart Object)

Related

Related

Related

<b>Usage</b>	<i>ChartObject.FullName</i>
<b>Description</b>	You can identify a chart's filename with or without its pathname. The <b>FullName</b> property of the Chart object returns the fully qualified pathname of the chart. (If the chart has not been saved, it returns the temporary name of the chart.) The <b>FullName</b> property is read only.
<b>Data Type</b>	String
<b>Value</b>	The fully qualified pathname of the chart
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FullName Property \(Application Object\)](#)

[Name Property \(Chart Object\)](#)

### Description

[Identifying a Chart's Filename](#)

### Object

[Chart Object](#)

**Related**

## FullName Property (Chart Object) Example

This example uses the **FullName** property of the Chart object to display the name and path of a chart.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.Save ("tst_chrt")                                ' Save the chart
Set Chart = ABC.ActiveChart                            ' Reset as active chart after save
MsgBox "Path name for this chart is " + Chart.FullName ' Display full path of file
```

## GuidelinesOn Property

Related

Related

Related

### Usage

*ChartObject.GuidelinesOn* = {True | False}

### Description

You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let you align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **GuidelinesOn** property to turn showing guidelines on and off. The **GuidelinesOn** property is read/write.

### Data Type

Integer (Boolean)

### Value

True shows guidelines; False does not.

### ABC Equivalent

The **GuidelinesOn** property is equivalent to clicking the Guidelines button at the bottom of the ABC FlowCharter window. Guidelines in the chart are displayed when the button is selected.

## Related Topics

### Language Elements

[AddHorizontalGuideline Method](#)

[AddVerticalGuideline Method](#)

[ClearGuidelines Method](#)

### Description

[Identifying a Chart's Filename](#)

### Object

[Chart Object](#)

## HasDiskFile Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> . <b>HasDiskFile</b>
<b>Description</b>	You use the <b>HasDiskFile</b> property to find if the chart has ever been saved to disk. The <b>HasDiskFile</b> property is read only.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the chart has been saved to disk; False means the chart is a new chart that has never been saved to disk.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[FullName Property \(Chart Object\)](#)

[Save Method](#)

[Saved Property](#)

### Description

[Saving Charts](#)

### Object

[Chart Object](#)

## Related

### HasDiskFile Property, Save Method Example

This example uses the **HasDiskFile** property and the **Save** method of the Chart object to check if a file has ever been saved, and save it if it has not.

```
Dim ABC As Object, Chart As Object, Obj1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                 ' Draw shape
Obj1.Text = "Unit 1"                                   ' Add text to shape

If Not Chart.HasDiskFile Then                           ' Has this file been saved?
    ' If not, save the file
    Chart.Save (InputBox$("Enter the file name"; "Save File"; ".af3"))
    ' If saved, display reminder
    Else MsgBox "Save your work often!", 48; "Don't Forget."
End If
```

## LaunchIndicator Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.LaunchIndicator = Indicator</i>
<b>Description</b>	The launch indicator appears on shapes with attached launches. You use the <b>LaunchIndicator</b> property to find or set the indicator, up to three characters, used for launched shapes. The <b>LaunchIndicator</b> property is read/write.
<b>Data Type</b>	String
<b>Value</b>	Text, up to three characters, that indicates that a shape in ABC is set to launch another application
<b>ABC Equivalent</b>	The <b>LaunchIndicator</b> property is equivalent to choosing Preferences in the File menu, clicking the Indicator Options button, and entering text in the Launch Indicator area.

## Related Topics

### Language Elements

[Launch Method](#)

[LaunchCommand Property](#)

[LaunchFlags Property](#)

[LaunchShadow Property](#)

[LaunchStartDir Property](#)

### Description

[Choosing Launch Indicators](#)

[Indicator Options](#)

### Object

[Chart Object](#)

## LaunchShadow Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.LaunchShadow</i> = {True   False}
<b>Description</b>	The <b>LaunchShadow</b> property lets you find or set whether shapes that have a launch associated with them show a shadow. The <b>LaunchShadow</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows a shadow on shapes that have a launch; False does not show it.
<b>ABC Equivalent</b>	The <b>LaunchShadow</b> property is equivalent to choosing Preferences in the File menu, clicking the Indicator Options button, and selecting or deselecting the Launch Shadow option.

## **Related Topics**

### **Language Elements**

[Launch Method](#)

[LaunchCommand Property](#)

[LaunchFlags Property](#)

[LaunchIndicator Property](#)

[LaunchStartDir Property](#)

### **Description**

[Choosing Launch Indicators](#)

[Indicator Options](#)

### **Object**

[Chart Object](#)

## Related

# LaunchShadow, LaunchIndicator Properties Example

This example uses the **LaunchShadow** property and the **LaunchIndicator** property of the Chart object to add a shadow to shapes with a launch and set the text indicator for shapes with a launch.

```
Dim ABC As Object, Chart As Object, Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.LaunchShadow = True                             ' Add shadow to shapes with a launch
Chart.LaunchIndicator = "!*"                          ' Change launch indicator string

Set Shape1 = Chart.DrawShape("Operation")              ' Draw a shape
Shape1.Shape.LaunchCommand = "c:\abc\abcdata.exe"     ' Set launch
```

## LinkIndicator Property

Related

Related

Related

### Usage

*ChartObject.LinkIndicator = Indicator*

### Description

The link indicator (up to three characters) appears on shapes with attached links. You use the **LinkIndicator** property to find or set the indicator used for linked shapes. The **LinkIndicator** property is read/write.

### Data Type

String

### Value

Text, up to three characters, that indicates that a shape in ABC is linked

### ABC Equivalent

The **LinkIndicator** property is equivalent to choosing Preferences in the File menu, clicking the Indicator Options button, and entering text in the Link Indicator area.

## Related Topics

### Language Elements

[IsLinked Property](#)

[Link Method](#)

[LinkedChartName Property](#)

[LinkFields Property](#)

[LinkShadow Property](#)

### Description

[Choosing Link Indicators](#)

[Indicator Options](#)

### Object

[Chart Object](#)

## Related

# LinkIndicator Property Example

This example uses the **LinkIndicator** property of the Chart object to set the link indicator for a chart.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Link1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.CloseAll                                           ' Close all charts
ABC.New                                                ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.LinkIndicator = "*L*"                            ' Set the link indicator to *L*
Set Shape1 = Chart.DrawShape("Decision")              ' Draw a shape
Set Link1 = Shape1.Shape.Link                         ' Link the shape to a new chart

MsgBox "Using the Window menu, switch to [CHART1] and notice that the shape is marked with '*L*'."

```

## LinkShadow Property

Related

Related

Related

### Usage

*ChartObject.LinkShadow* = {True | False}

### Description

The **LinkShadow** property lets you find or set whether shapes that have linked files show a shadow. The **LinkShadow** property is read/write.

### Data Type

Integer (Boolean)

### Value

True shows a shadow on shapes that have a linked file; False does not.

### ABC Equivalent

The **LinkShadow** property is equivalent to choosing Preferences in the File menu, clicking the Indicator Options button, and selecting or deselecting the Link Shadow option.

## Related Topics

### Language Elements

[IsLinked Property](#)

[Link Method](#)

[LinkedChartName Property](#)

[LinkFields Property](#)

[LinkIndicator Property](#)

### Description

[Choosing Link Indicators](#)

[Line Options](#)

### Object

[Chart Object](#)

## MasterItems Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> . <b>MasterItems</b>
<b>Description</b>	The <b>MasterItems</b> property lets you find the MasterItems objects. The <b>MasterItems</b> property is read only, but all the properties from the object it returns are read/write.
<b>Data Type</b>	Object
<b>Value</b>	The MasterItems objects
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[ChartNameShown Property](#)

### Description

[Displaying Master Items](#)

### Object

[Chart Object](#)

## Name Property (Chart Object)

Related

Related

Related

**Usage** *ChartObject.Name*

**Description** You use the **Name** property to return the name of the Chart object without the path. The **Name** property is read only.

**Data Type** String

**Value** The name of the Chart object without the path

**ABC Equivalent** None

## Related Topics

### Language Elements

- [Activate Method \(Chart Object\)](#)
- [ActiveChart Property](#)
- [Application Property](#)
- [Count Property](#)
- [Item Method \(Charts Collection\)](#)
- [Name Property \(Application Object\)](#)
- [Name Property \(FieldTemplate Object\)](#)
- [Name Property \(FieldValue Object\)](#)
- [Name Property \(Font Object\)](#)

### Description

- [Identifying a Chart's Filename](#)

### Object

- [Chart Object](#)

**Related**

## Name Property (Chart Object) Example

This example uses the **Name** property of the Chart object to display the name of a chart.

```
Sub Command1_Click ()
    Dim ABC As Object, Chart As Object

    Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
    ABC.Visible = True                                ' Make ABC visible
    Set Chart = ABC.ActiveChart                        ' Get the active chart

    Chart.Save ("tst_chrt")                            ' Save the chart

    MsgBox "File name for this chart is " + Chart.Name  ' Display file name
End Sub
```

## NextNumber Property

Related

Related

Related

### Usage

*ChartObject.NextNumber = NextChartNumber*

### Description

You use the **NextNumber** property to find or set the number for the next shape that is drawn. You can use various numbering systems, such as 1, 2, 3, or 1.1, 1.2, 1.3, or even text strings. The number is kept in the **NextNumber** property as a text string, because the number can contain text as well as numbers. The **NextNumber** property is incremented automatically each time you draw a shape. If **NextNumber** contains text with a number, the text remains and the number is incremented. For example; "Step 5" becomes "Step 6" when a new shape is drawn. If **NextNumber** contains only text, the text remains without incrementing.

The **Number** property of the Shape object contains the actual shape number for a particular shape. When you draw a shape, the value in the chart's **NextNumber** property is stored in the **Number** property, and the **NextNumber** property is incremented. You can change a shape's number by changing the value of the shape's **Number** property. The **NextNumber** property is read/write.

### Data Type

String

### Value

The number for the next shape drawn

### ABC Equivalent

The **NextNumber** property is equivalent to clicking the Renumber button and entering the number for the next shape in the Next Number text box in the ribbon.

## Related Topics

### Language Elements

[Number Property](#)

[NumberShown Property](#)

[Renumber Method](#)

### Description

[Numbering Shapes](#)

### Object

[Chart Object](#)

## Related

### NextNumber Property Example

This example uses the **NextNumber** property of the Chart object to set the number to be used by the next shape.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                 ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")

Chart.NextNumber = "100"                               ' Set number for next shape drawn

Set Obj1 = Chart.DrawShape("Operation")                 ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")
```

## NoRepaint Property

Related

Related

Related

### Usage

*ChartObject.NoRepaint* = {True | False}

### Description

The **NoRepaint** property lets you omit drawing each action. With the **NoRepaint** property set to True, you can have a 15% to 20% increase in speed. After the actions are complete, you update the screen using the **Repaint** method. Be sure to set the **NoRepaint** property to False when the program finishes drawing. The **NoRepaint** property is read/write.

### Data Type

Integer (Boolean)

### Value

True means to omit drawing each action; False means to draw each action.

### ABC Equivalent

None

## **Related Topics**

### **Language Elements**

[Repaint Method](#)

### **Description**

[Speeding Actions](#)

### **Object**

[Chart Object](#)

## NotIndicator Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.NotIndicator = IndicatorText</i>
<b>Description</b>	The <b>NotIndicator</b> property lets you find or set the text, up to three characters, that indicates that a shape in ABC has a note attached to it. The <b>NotIndicator</b> property is read/write.
<b>Data Type</b>	String
<b>Value</b>	Text, up to three characters, that indicates that a shape in ABC has a note attached to it
<b>ABC Equivalent</b>	The <b>NotIndicator</b> property is equivalent to choosing Preferences in the File menu, clicking the Indicator Options button, and entering text in the Note Symbol text box.

## **Related Topics**

### **Language Elements**

[NoteShadow Property](#)

[NoteText Property](#)

[NoteViewerVisible Property](#)

[NumberFont Property](#)

### **Description**

[Choosing Note Indicators](#)

[Indicator Options](#)

### **Object**

[Chart Object](#)

## NoteShadow Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.NoteShadow</i> = {True   False}
<b>Description</b>	You use the <b>NoteShadow</b> property to find or set whether shapes that have notes have a shadow. The <b>NoteShadow</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means shapes with notes have a shadow; False means they do not.
<b>ABC Equivalent</b>	The <b>NoteShadow</b> property is equivalent to choosing Preferences in the File menu, clicking the Alignment button, and selecting the Note Shadow option (True) or deselecting the option (False).

## **Related Topics**

### **Language Elements**

[NoteIndicator Property](#)

[NoteText Property](#)

[NoteViewerVisible Property](#)

[NumberFont Property](#)

### **Description**

[Choosing Note Indicators](#)

[Indicator Options](#)

### **Object**

[Chart Object](#)

## Related

### NoteShadow, NoteIndicator Properties Example

This example uses the **NoteShadow** property and **NoteIndicator** property of the Chart object to add a shadow to shapes with a note and set the text indicator for shapes with a note.

```
Dim ABC As Object, Chart As Object, Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

Chart.NoteShadow = True                                ' Apply shadow to shapes with notes
Chart.NoteIndicator = "*N*"                            ' Change note indicator string

Set Shape1 = Chart.DrawShape("Operation")              ' Draw a shape
Shape1.Shape.NoteText = "Check with Production"        ' Set note text for shape
```

## NumberFont Property

Related

Related

Related

### Usage

*ChartObject*.**NumberFont**

### Description

The **NumberFont** property lets you find or set properties for the Font object for shape numbers in a chart. All the properties of the Font object, such as bold and italic, are available through the **NumberFont** property. The **NumberFont** property is read only, but all the properties from the object it returns are read/write.

### Data Type

Object

### Value

A Font object

### ABC Equivalent

The **NumberFont** property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and setting the font attributes in the number font area.

## Related Topics

### Language Elements

[Bold Property](#)

[Color Property \(Font Object\)](#)

[Italic Property](#)

[LaunchIndicator Property](#)

[LaunchShadow Property](#)

[LinkIndicator Property](#)

[LinkShadow Property](#)

[Name Property \(Font Object\)](#)

[NoteIndicator Property](#)

[NoteShadow Property](#)

[ShowNodesOnLines Property](#)

[Size Property](#)

[Strikethrough Property](#)

[Underline Property](#)

### Description

[Indicator Options](#)

[Formatting Shape Numbers](#)

### Object

[Chart Object](#)

## Related

# NumberFont Property Example

This example uses the **NumberFont** property of the Chart object to set the attributes for the font used for numbers.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim ChartNumberFont As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set ChartNumberFont = Chart.NumberFont                 ' Set number font object

ChartNumberFont.Name = "Roman"                         ' Set font attributes
ChartNumberFont.Bold = True
ChartNumberFont.Italic = True

Set Obj1 = Chart.DrawShape("Operation")                ' Draw shapes
Set Obj2 = Chart.DrawShape("Decision")
```

## Objects Property

Related

Related

Related

### Usage

*ChartObject*.**Objects**

### Description

The **Objects** property lets you find the objects included in the Objects collection. The **Objects** property is read only, but all the properties from the object it returns are read/write.

### Data Type

Collection object

### Value

The objects included in the Objects collection

### ABC Equivalent

None

## Related Topics

### Language Elements

[ObjectType Property](#)

### Description

[Identifying an Object](#)

### Object

[Chart Object](#)

## PageCount Property

Related

Related

Related

**Usage** *ChartObject.PageCount = Number*

**Description** You use the **PageCount** property to find the number of pages in the chart, including pages with no objects on them. The **PageCount** property is read only.

**Data Type** Integer

**Value** The number of pages in the chart

**ABC Equivalent** None

## Related Topics

### Language Elements

[ScrollLeft Property](#)

[ScrollPage Method](#)

[ScrollPosition Method](#)

[ScrollTop Property](#)

[View Property](#)

### Description

[Adjusting the Page Layout](#)

### Object

[Chart Object](#)

## Related

# PageCount Property Example

This example uses the **PageCount** property of the Chart object to show the number of used pages in a chart.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

For X = 1 To 10                                         ' Draw shapes
    Set Shape = Chart.DrawShape("Decision")
Next X

Chart.View = 2                                          ' View used pages

MsgBox "Chart pages: " + Chart.PageCount + "."         ' Number of pages in the chart
```

## PageLayout Property

Related

Related

Related

Related

### Usage

*ChartObject*.**PageLayout**

### Description

You use the **PageLayout** property to find the PageLayout object. The **PageLayout** property is read only, but the properties from the object it returns are read/write.

### Data Type

Collection object

### Value

The PageLayout object

### ABC Equivalent

None

## Related Topics

### Description

[Adjusting the Page Layout](#)

### Object

[Chart Object](#)

## Protected Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> . <b>Protected</b>
<b>Description</b>	You use the <b>Protected</b> property to find whether a chart is protected. The <b>Protected</b> property is read only.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means that the chart is password protected; False means that the chart is not password protected.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[SetProtection Method](#)

### Description

[Protecting Charts](#)

### Object

[Chart Object](#)

**Related**

## Protected Property Example

This example uses the **Protected** property of the Chart object to give a chart a password.

```
Dim ABC As Object, Chart As Object, Shape As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

If NOT Chart.Protected Then                             ' Set password
    Chart.SetProtection 1,"gipper"
End If
```

## ReadOnly Property

Related

Related

Related

**Usage** *ChartObject.ReadOnly*

**Description** You can determine whether a chart is read only by using the **ReadOnly** property of the Chart object. Read-only charts cannot be saved under the same filename. The **ReadOnly** property is read only.

**Data Type** Integer (Boolean)

**Value** True means that the chart is read only; False means that the chart is read/write.

**ABC Equivalent** None

## Related Topics

### Description

[Read Only Charts](#)

### Object

[Chart Object](#)

**Related**

## ReadOnly Property Example

This example uses the **ReadOnly** property of the Chart object to determine if a chart was opened as read only.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible

Set Chart = ABC.Open("C:\ABC\TUTORIAL\COLOR.AF3", 1) ' Open file read only

If Chart.ReadOnly Then                             ' Is chart read only?
    MsgBox "This file has the read-only attribute."
End If
```

## Saved Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.Saved</i>
<b>Description</b>	The <b>Saved</b> property determines if the Chart object in memory is the same as on disk. The <b>Saved</b> property is read only.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means that the chart in memory is the same as the chart file on disk; False means that the chart in memory is not the same as the chart file on disk.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Save Method](#)

### Description

[Saving Charts](#)

### Object

[Chart Object](#)

## ScrollLeft Property

Related

Related

Related

**Usage** *ChartObject.ScrollLeft = Distance*

**Description** You use the **ScrollLeft** property to find or set the left point visible in the chart. The **ScrollLeft** property is read/write.

**Data Type** Double

**Value** The left point visible in the chart

**ABC Equivalent** The **ScrollLeft** property is equivalent to clicking the horizontal scroll bar.

## Related Topics

### Language Elements

[PageCount Property](#)  
[ScrollPage Method](#)  
[ScrollPosition Method](#)  
[ScrollTop Property](#)  
[View Property](#)

### Description

[Viewing a Chart](#)

### Object

[Chart Object](#)

## ScrollTop Property

Related

Related

Related

**Usage** *ChartObject.ScrollTop = Distance*

**Description** You use the **ScrollTop** property to find or set the top point visible in the chart. The **ScrollTop** property is read/write.

**Data Type** Double

**Value** The top point visible in the chart

**ABC Equivalent** The **ScrollTop** property is equivalent to clicking the vertical scroll bar.

## Related Topics

### Language Elements

[PageCount Property](#)  
[ScrollPage Method](#)  
[ScrollPosition Method](#)  
[ScrollLeft Property](#)  
[View Property](#)

### Description

[Viewing a Chart](#)

### Object

[Chart Object](#)

## SelectedLineCount Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.SelectedLineCount</i>
<b>Description</b>	The <b>SelectedLineCount</b> property lets you find the number of lines in the Chart object. The <b>SelectedLineCount</b> property contains the number of selected lines, not the number of selected line segments, so the routing of the lines does not affect the count. The <b>SelectedLineCount</b> property is read only.
<b>Data Type</b>	Integer
<b>Value</b>	The number of lines in the Chart object
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Count Property](#)

[SelectedObjectCount Property](#)

[SelectedOtherCount Property](#)

[SelectedShapeCount Property](#)

### Description

[Finding the Total Number of Objects](#)

### Object

[Chart Object](#)

## Related

### SelectedLineCount Property Example

This example uses the **SelectedLineCount** property of the Chart object to find the number of lines that are selected in a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Obj3 As Object, Line1 As Object, Line2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Terminal")                 ' Draw shapes
Set Obj2 = Chart.DrawShape("Operation")
Set Obj3 = Chart.DrawShape("Decision")

Set Line1 = Chart.DrawLine(Obj1, Obj2)                 ' Draw lines
Set Line2 = Chart.DrawLine(Obj2, Obj3)
Chart.Select (2)                                       ' Select all objects

MsgBox "There are " + Chart.SelectedLineCount + " line(s) selected in the chart."
```

## SelectedObjectCount Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> . <b>SelectedObjectCount</b>
<b>Description</b>	The <b>SelectedObjectCount</b> property lets you find the number of selected objects in the Chart object. It equals the sum of the values of the <b>SelectedShapeCount</b> , <b>SelectedLineCount</b> , and <b>SelectedOtherCount</b> properties. The <b>SelectedObjectCount</b> property is read only.
<b>Data Type</b>	Integer
<b>Value</b>	The number of selected objects in the Chart object
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Count Property](#)

[SelectedLineCount Property](#)

[SelectedOtherCount Property](#)

[SelectedShapeCount Property](#)

### Description

[Finding the Total Number of Objects](#)

### Object

[Chart Object](#)

**Related**

## SelectedObjectCount Property Example

This example uses the **SelectedObjectCount** property of the Chart object to display the number of selected objects in a chart.

```
Sub Command1_Click ()
    Dim ABC As Object, Chart As Object, Obj1 As Object
    Dim X As Integer

    Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
    ABC.Visible = True                                ' Make ABC visible
    Set Chart = ABC.ActiveChart                        ' Get the active chart

    Set Obj1 = Chart.DrawShape("Operation")           ' Draw Operation shape

    For X = 1 To 3                                     ' Duplicate shape three times
        Chart.Duplicate
    Next X

    Chart.Select (2)                                   ' Select all objects

    MsgBox "There are " + Chart.SelectedObjectCount + " objects selected"
End Sub
```

## SelectedOtherCount Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.SelectedOtherCount</i>
<b>Description</b>	The <b>SelectedOtherCount</b> property lets you find the number of objects in the Chart object that are not shapes or lines. It includes master item objects such as the date and headers, OLE objects, bitmaps, and other objects pasted into ABC. The <b>SelectedOtherCount</b> property is read only.
<b>Data Type</b>	Integer
<b>Value</b>	The number of objects in the Chart object that are not shapes or lines
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Count Property](#)

[SelectedLineCount Property](#)

[SelectedObjectCount Property](#)

[SelectedShapeCount Property](#)

### Description

[Finding the Total Number of Objects](#)

### Object

[Chart Object](#)

## Related

### SelectedOtherCount Property Example

This example uses the **SelectedOtherCount** property of the Chart object to find the number objects other than shapes and lines that are selected in a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Obj3 As Object, Line1 As Object, Line2 As Object
Dim Text1 As Object, Text2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Terminal")                 ' Draw shapes
Set Obj2 = Chart.DrawShape("Operation")
Set Obj3 = Chart.DrawShape("Decision")

Set Line1 = Chart.DrawLine(Obj1, Obj2)                ' Draw lines
Set Line2 = Chart.DrawLine(Obj2, Obj3)

Chart.DrawPositionX = 2                               ' Set draw position
Chart.DrawPositionY = 2.5

Set Text1 = Chart.DrawTextBlock("ABC FlowCharter")    ' Draw text objects
Set Text2 = Chart.DrawTextBlock("OLE2 Automation")

Chart.Select (2)                                       ' Select all objects

MsgBox "There are " + Chart.SelectedOtherCount + " items(s) selected in the chart other than
lines or shapes"
```

## SelectedShapeCount Property

Related

Related

Related

**Usage** *ChartObject.SelectedShapeCount*

**Description** The **SelectedShapeCount** property lets you find the number of selected shapes in the Chart object. The **SelectedShapeCount** property is read only.

**Data Type** Integer

**Value** The number of selected shapes in the Chart object

**ABC Equivalent** None

## Related Topics

### Language Elements

[Count Property](#)

[DeselectAll Method](#)

[Select Method](#)

[Selected Property](#)

[SelectedLineCount Property](#)

[SelectedObjectCount Property](#)

[SelectedOtherCount Property](#)

### Description

[Finding the Total Number of Objects](#)

[Selecting Shapes](#)

### Object

[Chart Object](#)

## Related

### SelectedShapeCount Property Example

This example uses the **SelectedShapeCount** property of the Chart object to find the number of shapes that are selected in a chart.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim Obj3 As Object, Line1 As Object, Line2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Terminal")                 ' Draw shapes
Set Obj2 = Chart.DrawShape("Operation")
Set Obj3 = Chart.DrawShape("Decision")

Set Line1 = Chart.DrawLine(Obj1, Obj2)                ' Draw lines
Set Line2 = Chart.DrawLine(Obj2, Obj3)
Chart.Select (2)                                       ' Select all objects

MsgBox "There are " + Chart.SelectedShapeCount + " shape(s) selected in the chart."
```

## ShowLegend Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.ShowLegend</i> = {True   False}
<b>Description</b>	The <b>ShowLegend</b> property lets you choose to show or hide the Legend. The Legend in the shows the accumulation of the data fields in a chart. The <b>ShowLegend</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the Legend; False hides the Legend.
<b>ABC Equivalent</b>	The <b>ShowLegend</b> property is equivalent to choosing Show Legend or Hide Legend in the ABC Fields menu.

## Related Topics

### Language Elements

[Accumulation Property](#)

[AccumulationMethod Property](#)

### Description

[Viewing the Legend](#)

### Object

[Chart Object](#)



## ShowNodesOnLines Property

Related

Related

Related

### Usage

*ChartObject.ShowNodesOnLines* = {True | False}

### Description

The **ShowNodesOnLines** property lets you find or set whether lines show connection nodes. Nodes appear where lines connect to each other. They help you distinguish between connected lines and lines that merely overlap. Nodes are represented by small, filled circles on lines. The **ShowNodesOnLines** property is read/write.

### Data Type

Integer (Boolean)

### Value

True means nodes are shown on lines; False means nodes are not shown on lines.

### ABC Equivalent

The **ShowNodesOnLines** property is equivalent to choosing Preferences in the File menu, clicking the Indicator Options button, and selecting the Show Nodes on Lines option (True) or deselecting the option (False).

## Related Topics

### Language Elements

[LaunchIndicator Property](#)

[LaunchShadow Property](#)

[LineCrossoverSize Property](#)

[LineCrossoverStyle Property](#)

[LinkIndicator Property](#)

[LinkShadow Property](#)

[NoteIndicator Property](#)

[NoteShadow Property](#)

[NumberFont Property](#)

### Description

[Displaying Nodes on Connecting Lines](#)

### Object

[Chart Object](#)



## Type Property (Chart Object)

Related

Related

Related

**Usage** *ChartObject.Type = ChartType*

**Description** The **Type** property of the Chart object lets you find or set a hidden string field up to eight characters in length indicating the chart type. This field is never used within ABC, but is useful within an ABC events VBX. The **Type** property is read/write.

**Data Type** String

**Value** The type of a chart. The default is "" (an empty string).

**ABC Equivalent** None

## Related Topics

### Language Elements

[Type Property \(FieldTemplate Object\)](#)

[Type Property \(FieldValue Object\)](#)

[Type Property \(Line Object\)](#)

[Type Property \(Object Object\)](#)

### Description

[Linking EXEs to Charts](#)

[Event Variables](#)

### Object

[Chart Object](#)

## Units Property (Chart Object)

Related

Related

Related

### Usage

*ChartObject.Units = UnitsIndicator*

### Description

You use the **Units** property of the Chart object to find or set the units for measurement in the Chart object and all its child chart objects. In addition, the **Units** property specifies the size and distance values passed in the Preferences object. Default unit value is 0 (inches) for each new Preferences object.

### Data Type

None

### Value

The units used for measurements are listed in the table below.

<b>UnitsIndicator</b>	<b>Description</b>
-----------------------	--------------------

0	Inches
---	--------

1	Centimeters
---	-------------

### ABC Equivalent

The **Units** property is equivalent to choosing PageLayout in the File menu and choosing the Inches or Centimeters option.

## **Related Topics**

### **Language Elements**

[Units Property \(Preferences Object\)](#)

### **Description**

[Defining Measurement Units for a Chart](#)

### **Object**

[Chart Object](#)

## Related

### Units Property (Chart Object) Example

This example uses the **Units** property of the Chart object to set the units for a chart.

```
Dim ABC As Object, Chart As Object
Dim ChartUnits As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

ChartUnits = Val(InputBox$("Enter 0 for inches" + Chr$(13) + "Enter 1 for centimeters", "Chart
Units"))                                               ' Get input; Chr$(13) is Carriage Return

Chart.Units = ChartUnits                               ' Set units
```

## View Property

Related

Related

Related

### Usage

*ChartObject.View = View*

### Description

You use the **View** property to find or set the view of the chart. The **View** property is read/write.

### Data Type

Integer

### Value

The value in the **View** property indicates the display page.

<b>View</b>	<b>Description</b>
-------------	--------------------

0	One to one
---	------------

1	Current page
---	--------------

2	Used pages
---	------------

3	Percentage zoom
---	-----------------

### ABC Equivalent

The **View** property is equivalent to clicking the View tool and clicking one of the buttons.

## Related Topics

### Language Elements

[PageCount Property](#)

[ScrollLeft Property](#)

[ScrollPage Method](#)

[ScrollPosition Method](#)

[ScrollTop Property](#)

[ZoomPercentage Property](#)

### Description

[Viewing a Chart](#)

### Object

[Chart Object](#)

## Related

# View Property Example

This example uses the **View** property of the Chart object to set the view of a chart.

```
Dim ABC As Object, Chart As Object
Dim String1 As String, String2 As String, String3 As String
Dim String4 As String, String5 As String
Dim ChartView As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Set Chart object

String1 = "Choose a View type:" + Chr$(13)            ' Create text for input
String2 = "0" + Chr$(9) + "OneToOne"                 ' Chr$(13) is Carriage Return
String3 = "1" + Chr$(9) + "CurrentPage"               ' Chr$(9) is Tab
String4 = "2" + Chr$(9) + "UsedPages"
String5 = String1 + Chr$(13) + String2 + Chr$(13) + String3 + Chr$(13) + String4

ChartView = Val(InputBox$(String5; "Chart View"))     ' Get input

Chart.View = ChartView                                ' Set view
```

## Activate Method (Chart Object)

Related

Related

Related

**Usage** *ChartObject.Activate*

**Description** You use the **Activate** method of the Chart object to pull the chart to the front of the ABC workspace. When multiple charts are open, this brings one to the front, or activates it.

**ABC Equivalent** The **Activate** method is equivalent to opening the Window menu and choosing the chart from the numbered list of open charts.

## Related Topics

### Language Elements

[Activate Method \(Application Object\)](#)

[ActiveChart Property](#)

[Application Property](#)

[Count Property](#)

[Item Method \(Charts Collection\)](#)

[Name Property \(Chart Object\)](#)

[Visible Property \(Application Object\)](#)

### Description

[Activating a Chart](#)

### Object

[Chart Object](#)



## AddHorizontalGuideline Method

Related

Related

Related

### Usage

*ChartObject.AddHorizontalGuideline Position*

The *Position* element specifies the vertical location of the guideline.

### Description

The ABC user can use guidelines to align objects. When dragging a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let the user align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **AddHorizontalGuideline** method to add a horizontal guideline at the vertical position passed.

### Data Type

Double

### Value

None

### ABC Equivalent

The **AddHorizontalGuideline** method is equivalent to dragging a guideline from the rulers.

## Related Topics

### Language Elements

[AddVerticalGuideline Method](#)

[ClearGuidelines Method](#)

[GuidelinesOn Property](#)

### Description

[Using Guidelines](#)

### Object

[Chart Object](#)

## Related

# AddHorizontalGuideline Method, AddVerticalGuideline Method, and GuidelinesOn Property Example

This example uses the **AddHorizontalGuideline** method, **AddVerticalGuideline** method, and **GuidelinesOn** property of the Chart object to position and show guidelines.

```
Dim ABC As Object, Chart As Object
```

```
Set ABC = CreateObject("ABCFlow.application")
```

```
ABC.Visible = True
```

```
Set Chart = ABC.ActiveChart
```

```
Chart.AddHorizontalGuideline (3)
```

```
Chart.AddVerticalGuideline (3)
```

```
Chart.GuidelinesOn = True
```

```
' Start ABC
```

```
' Make ABC visible
```

```
' Get the active chart
```

```
' Place horizontal guideline at 3 units
```

```
' Place vertical guideline at 3 units
```

```
' Show guidelines
```

## AddVerticalGuideline Method

Related

Related

Related

### Usage

*ChartObject.AddVerticalGuideline Position*

The *Position* element specifies the horizontal location of the guideline.

### Description

You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let you align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **AddVerticalGuideline** method to add a vertical guideline at the horizontal position passed.

### Data Type

Double

### Value

None

### ABC Equivalent

The **AddVerticalGuideline** method is equivalent to dragging a guideline from the rulers.

## Related Topics

### Language Elements

[AddHorizontalGuideline Method](#)

[ClearGuidelines Method](#)

[GuidelinesOn Property](#)

### Description

[Using Guidelines](#)

### Object

[Chart Object](#)

## Clear\_ Method

Related

Related

Related

Related

### Usage

*ChartObject*.Clear\_  
*ObjectObject*.Clear\_

### Description

You use the **Clear\_** method of the Chart object to clear (delete) all selected objects. You use the **Clear\_** method of the Object object to delete the object object. This is useful in removing a temporary object created as part of a routine using the **SetDefaults** method.

### Data Type

Integer (Boolean)

### Value

True means the deletion was successful; False means the deletion was not successful.

### ABC Equivalent

The **Clear\_** method is equivalent to pressing the **Del** key or choosing Clear in the ABC Edit menu.

## Related Topics

### Language Elements

[DeselectAll Method](#)

[Select Method](#)

[Selected Property](#)

[SelectShapeType Method](#)

[SetDefaults Method](#)

### Description

[Clearing Selected Objects](#)

[Speeding Actions](#)

### Object

[Chart Object](#)

**Related**

## Clear\_ Method Example

This example uses the **Clear\_** method of the Chart object to find and delete the selected objects.

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

For X = 1 To 3
    Set Obj1 = Chart.DrawShape("Document")
    Set Obj2 = Chart.DrawShape("Decision")
Next X

Chart.SelectShapeType "Decision"

Chart.Clear_
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Draw shapes  
  
' Select all Decision shapes  
' Delete selected objects

## ClearGuidelines Method

Related

Related

Related

### Usage

*ChartObject*.**ClearGuidelines**

### Description

You can use guidelines to align objects. When you drag a shape near a guideline, the shape's sides or center snap into alignment with the guideline if the Align to Rulers option is selected in the Preferences dialog box. Guidelines let you align shapes of different sizes for an attractive, organized look. The guidelines do not appear in the printed chart. You use the **ClearGuidelines** property to delete all guidelines from the chart. There is currently no way to remove a single guideline.

### ABC Equivalent

The **ClearGuidelines** method is equivalent to dragging all guidelines from the chart back into the rulers.

## Related Topics

### Language Elements

[AddHorizontalGuideline Method](#)

[AddVerticalGuideline Method](#)

[GuidelinesOn Property](#)

### Description

[Using Guidelines](#)

### Object

[Chart Object](#)



## DeselectAll Method

Related

Related

Related

**Usage** *ChartObject.DeselectAll*

**Description** You use the **DeselectAll** method to deselect all objects. The **DeselectAll** method has the same effect as the Select method with a value of 3.

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[Clear Method](#)

[Select Method](#)

[Selected Property](#)

[SelectShapeType Method](#)

### **Description**

[Selecting Objects in a Chart](#)

[Selecting Shapes](#)

### **Object**

[Chart Object](#)



## CloseChart Method

Related

Related

Related

**Usage** *ChartObject.CloseChart*

**Description** You use the **CloseChart** method to close the Chart object without any prompt to save the chart.

**ABC Equivalent** The **CloseChart** method is equivalent to choosing Close in the File menu, except that there is not a prompt to change a saved chart.

## **Related Topics**

### **Language Elements**

[CloseAll Method](#)

[Save Method](#)

### **Description**

[Closing Charts](#)

### **Object**

[Chart Object](#)

## Copy Method

Related

Related

Related

**Usage** *ChartObject.Copy*

**Description** You use the **Copy** method to copy selected objects to the Windows Clipboard.

**Data Type** Integer (Boolean)

**Value** True means the copy was successful; False means the copy was not successful.

**ABC Equivalent** The **Copy** method is equivalent to choosing Copy in the ABC Edit menu.

## **Related Topics**

### **Language Elements**

[Cut Method](#)

[Duplicate Method \(Chart Object\)](#)

[Paste Method](#)

[PasteSpecial Method](#)

### **Description**

[Cutting, Copying, and Pasting Objects](#)

### **Object**

[Chart Object](#)

## Cut Method

Related

Related

Related

**Usage** *ChartObject.Cut*

**Description** You use the **Cut** method to cut selected objects to the Windows Clipboard.

**Data Type** Integer (Boolean)

**Value** True means the cut was successful; False means the cut was not successful.

**ABC Equivalent** The **Cut** method is equivalent to choosing Cut in the ABC Edit menu.

## **Related Topics**

### **Language Elements**

[Copy Method](#)

[Paste Method](#)

[PasteSpecial Method](#)

### **Description**

[Cutting, Copying, and Pasting Objects](#)

### **Object**

[Chart Object](#)



## DrawFreeLine Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.DrawFreeLine (HorizontalLocation, VerticalLocation)</i> The <i>HorizontalLocation</i> element is the X location of the end point of the line. The <i>VerticalLocation</i> element is the Y location of the end point of the line.
<b>Description</b>	You use the <b>DrawFreeLine</b> method to draw an unconnected line from the current chart position to a specified end point. The X and Y positions are measured from the top left corner of the chart page. The line is not selected.
<b>Data Type</b>	Object
<b>Value</b>	The method returns the line object that is drawn. Both elements are doubles.
<b>ABC Equivalent</b>	The <b>DrawFreeLine</b> method is equivalent to drawing a line not connected to any shapes.

## Related Topics

### Language Elements

[DrawLine Method](#)

[DrawLineToOneObject Method](#)

[DrawPositionX Property](#)

[DrawPositionY Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Setting the Current Drawing Position](#)

[Drawing Unconnected Lines](#)

### Object

[Chart Object](#)

## Related

### DrawFreeLine Method, CurrentLineRouting Property Example

This example uses the **CurrentLineRouting** property and **DrawFreeLine** method of the Chart object to set the line routing and draw a line.

```
Dim ABC As Object, Chart As Object, Obj1 As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                 ' Draw a shape

Chart.CurrentLineRouting = 2                           ' Set routing to curve
Set Line1 = Chart.DrawFreeLine(4, 3)                   ' Draw a line from the current position
```

## DrawLine Method

Related

Related

Related

<b>Usage</b>	<p><i>ChartObject.DrawLine (ShapeObject1, ShapeObject2 [, ExitDirection] [, EnterDirection])</i></p> <p>The <i>ShapeObject1</i> element is the first shape that the line is connected to. The <i>ShapeObject2</i> element is the second shape that the line is connected to. The <i>ExitDirection</i> element, which is optional, specifies the side where the line exits the first shape. The <i>EnterDirection</i> element, which is optional, specifies the side where the line enters the second shape.</p>										
<b>Description</b>	<p>You use the <b>DrawLine</b> method to draw lines that connect two shapes. You specify the two shapes you want to connect and, optionally, the sides of the shapes that the line connects to. The line is not selected.</p>										
<b>Data Type</b>	<p>Object. The <i>ShapeObject1</i> element and <i>ShapeObject2</i> element are Shape objects. The <i>ExitDirection</i> element and <i>EnterDirection</i> element, which are optional, are integers.</p>										
<b>Value</b>	<p>The new Line_ object. The following chart describes the direction values.</p> <table><thead><tr><th>Value</th><th>Direction</th></tr></thead><tbody><tr><td>0</td><td>North</td></tr><tr><td>1</td><td>East</td></tr><tr><td>2</td><td>South</td></tr><tr><td>3</td><td>West</td></tr></tbody></table>	Value	Direction	0	North	1	East	2	South	3	West
Value	Direction										
0	North										
1	East										
2	South										
3	West										
<b>ABC Equivalent</b>	<p>The <b>DrawLine</b> property is equivalent to drawing a line from one shape to another.</p>										

## Related Topics

### Language Elements

[DrawFreeLine Method](#)

[DrawLineToOneObject Method](#)

### Description

[Drawing Lines that Connect Shapes](#)

### Object

[Chart Object](#)



## DrawLineToOneObject Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.DrawLineToOneObject</i> ( <i>ShapeObject</i> [, <i>EnterDirection</i> ]) The <i>ShapeObject</i> element is the shape that the line is connected to. The <i>EnterDirection</i> element, which is optional, specifies the side where the line enters the shape.										
<b>Description</b>	You use the <b>DrawLineToOneObject</b> method to draw a line from the current chart position to a specified shape. The line starts at the chart's current drawing position and ends at the shape you specify with <i>ShapeObject</i> . You can optionally specify the side of the shape that the line connects to. The line is not selected.										
<b>Data Type</b>	Object. The <i>ShapeObject</i> element is a Shape object. The <i>EnterDirection</i> element, which is optional, is an integer.										
<b>Value</b>	The method returns the line object that is drawn. The following table shows the values of the <i>EnterDirection</i> element and their meanings. <table><thead><tr><th>Value</th><th>Direction</th></tr></thead><tbody><tr><td>0</td><td>North</td></tr><tr><td>1</td><td>East</td></tr><tr><td>2</td><td>South</td></tr><tr><td>3</td><td>West</td></tr></tbody></table>	Value	Direction	0	North	1	East	2	South	3	West
Value	Direction										
0	North										
1	East										
2	South										
3	West										
<b>ABC Equivalent</b>	The <b>DrawLineToOneObject</b> property is equivalent to drawing a line to a shape.										

## Related Topics

### Language Elements

[DrawFreeLine Method](#)

[DrawLine Method](#)

### Description

[Setting the Current Drawing Position](#)

[Drawing Lines to One Shape](#)

### Object

[Chart Object](#)



## DrawShape Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.DrawShape</i> ([ <i>ShapeName</i> ]) The <i>ShapeName</i> element is the optional name of the shape.
<b>Description</b>	You use the <b>DrawShape</b> method to draw shapes. The line is not selected. By default, the <b>DrawShape</b> method uses the current shape in the Shape Palette. You can optionally specify the name of the shape you want to draw. All of the shape palettes that ship with ABC have names that appear in the hint line or in the bubble help when the mouse pauses over them. In ABC, the shape's name is defined in the Shape Properties dialog box. (See the documentation that ships with ABC for more information on the available palettes and shapes.) You can open the Shape Properties dialog box by choosing Shape Properties in the Palette menu of the Shape Palette. Shapes are automatically placed at the chart's current drawing position.
<b>Data Type</b>	Object. The <i>ShapeName</i> element is a string.
<b>Value</b>	The new object or Null if the creation failed
<b>ABC Equivalent</b>	The DrawShape method is equivalent to clicking the Shape Tool, choosing the shape you want in the Shape Palette, and clicking in the drawing area.

## Related Topics

### Language Elements

[CurrentShape Property](#)  
[DrawPositionX Property](#)  
[DrawPositionY Property](#)  
[DrawSpacingX Property](#)  
[DrawSpacingY Property](#)

### Description

[Drawing Shapes](#)

### Object

[Chart Object](#)

## DrawTextBlock Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.DrawTextBlock (TextString)</i> The <i>TextString</i> element is the text you want to create.
<b>Description</b>	You use the <b>DrawTextBlock</b> method to create a text block. The text appears at the current drawing position. The text is not selected.
<b>Data Type</b>	Object
<b>Value</b>	The text block that is drawn
<b>ABC Equivalent</b>	The <b>DrawTextBlock</b> method is equivalent to clicking the Text tool, positioning the cursor, and typing text.

## Related Topics

### Language Elements

[DrawPositionX Property](#)

[DrawPositionY Property](#)

### Description

[Setting the Current Drawing Position](#)

[Moving Objects](#)

[Creating Text Blocks](#)

### Object

[Chart Object](#)

## Related

### DrawTextBlock Method Example

This example uses the **DrawTextBlock** method of the Chart object to create text objects.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object, Text2 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

Chart.DrawPositionX = 2                                ' Set draw position
Chart.DrawPositionY = 2.5

Set Text1 = Chart.DrawTextBlock("ABC FlowCharter")     ' Draw text objects
Set Text2 = Chart.DrawTextBlock("OLE2 Automation")
```

## Duplicate Method (Chart Object)

Related

Related

Related

<b>Usage</b>	<i>ChartObject.Duplicate</i>
<b>Description</b>	You use the <b>Duplicate</b> method of the Chart object to create a duplicate of the selected objects. The newly created objects will be the only selected objects in the Chart after you call this method.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the duplication was successful; False means the duplication was not successful
<b>ABC Equivalent</b>	None

## **Related Topics**

### **Language Elements**

[Copy Method](#)

[Duplicate Method \(Object Object\)](#)

[Paste Method](#)

### **Description**

[Duplicating Objects](#)

### **Object**

[Chart Object](#)

**Related**

## Duplicate Method (Chart Object) Example

This example uses the **Duplicate** method of the Chart object to create duplicates of selected shapes.

```
Dim ABC As Object, Chart As Object, Obj1 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")                ' Draw Operation shape
Obj1.Text = "Unit 1"                                   ' Add text to shape

Chart.Select (0)                                       ' Select all shapes

For X = 1 To 3                                         ' Duplicate shape three times
    Chart.Duplicate
Next X
```

## InsertObjectFromFile Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.InsertObjectFromFile</i> ( <i>Filename</i> [, <i>AsIcon</i> ] [, <i>AsLink</i> ]) The <i>Filename</i> element lets you specify the file to insert. The <i>AsIcon</i> element lets you paste the file as an icon. The <i>AsLink</i> element lets you paste the file linked.
<b>Description</b>	You use the <b>InsertObjectFromFile</b> method to insert a new OLE client object from a file into your chart. You can optionally add the element <i>AsIcon</i> to paste the file as an icon or the element <i>AsLink</i> to paste the file linked. The method returns the file that is inserted as an object.
<b>Data Type</b>	Object. The <i>Filename</i> element is a string. The <i>AsIcon</i> element and <i>AsLink</i> element are integers (Boolean).
<b>Value</b>	The object that was inserted
<b>ABC Equivalent</b>	The <b>InsertObjectFromFile</b> method is equivalent to choosing Insert Object in the ABC Edit menu, choosing the Create from File option, selecting the file you want to insert, and clicking OK. The <i>AsIcon</i> element is equivalent to selecting the Display As Icon option. The <i>AsLink</i> element is equivalent to selecting the Link to File option.

## **Related Topics**

### **Language Elements**

[DoVerb Method](#)

[ObjectType Property](#)

[PasteLink Method](#)

[UpdateFields Method](#)

### **Description**

[Using OLE Client Objects](#)

### **Object**

[Chart Object](#)

**Related**

## InsertObjectFromFile Method Example

This example uses the **InsertObjectFromFile** method of the Chart object to insert a sound from a file.

```
Dim ABC As Object, Chart As Object
Dim objOLE As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

' Insert OLE object
Set objOLE = Chart.InsertObjectFromFile("C:\WINDOWS\TADA.WAV", True, True)
```

## Paste Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.Paste</i> ( <i>HorizontalLocation</i> [, <i>VerticalLocation</i> ]) The <i>HorizontalLocation</i> element is the horizontal location of the paste. The <i>VerticalLocation</i> element is the vertical location of the paste.
<b>Description</b>	You use the <b>Paste</b> method to paste selected objects from the Windows Clipboard. You can optionally specify a horizontal and vertical location for the paste. You set the units used for the location using the <b>Units</b> property.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the paste was successful; False means the paste was not successful.
<b>ABC Equivalent</b>	The <b>Paste</b> method is equivalent to choosing Paste in the ABC Edit menu.

## Related Topics

### Language Elements

[Copy Method](#)

[Cut Method](#)

[Duplicate Method \(Chart Object\)](#)

[PasteSpecial Method](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Cutting, Copying, and Pasting Objects](#)

### Object

[Chart Object](#)

## PasteSpecial Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.PasteSpecial</i> ( <i>Format</i> [, <i>AsIcon</i> ] [, <i>HorizontalLocation</i> ] [, <i>VerticalLocation</i> ]) The <i>Format</i> element lets you specify the format to use for the paste. The optional <i>AsIcon</i> element lets you paste the Clipboard contents as an icon. The optional <i>HorizontalLocation</i> element is the horizontal location of the paste. The optional <i>VerticalLocation</i> element is the vertical location of the paste.																				
<b>Description</b>	You use the <b>PasteSpecial</b> method to paste selected objects from the Windows Clipboard specifying a format. You can optionally specify that the object be pasted as an icon. You can optionally specify a horizontal and vertical location for the paste. You set the units for the location using the <b>Units</b> property.																				
<b>Data Type</b>	Integer (Boolean)																				
<b>Value</b>	True means the paste was successful; False means the paste was not successful. The values for the formats are in the following table. <table><thead><tr><th>Value</th><th>Format</th></tr></thead><tbody><tr><td>0</td><td>ABC Native</td></tr><tr><td>1</td><td>OLE Client Embed</td></tr><tr><td>2</td><td>ABC Rich Text</td></tr><tr><td>3</td><td>Rich Text Format (RTF)</td></tr><tr><td>4</td><td>Unformatted Text</td></tr><tr><td>5</td><td>Metafile</td></tr><tr><td>6</td><td>Device-Independent Bitmap</td></tr><tr><td>7</td><td>Bitmap</td></tr><tr><td>8</td><td>OLE Client Link</td></tr></tbody></table>	Value	Format	0	ABC Native	1	OLE Client Embed	2	ABC Rich Text	3	Rich Text Format (RTF)	4	Unformatted Text	5	Metafile	6	Device-Independent Bitmap	7	Bitmap	8	OLE Client Link
Value	Format																				
0	ABC Native																				
1	OLE Client Embed																				
2	ABC Rich Text																				
3	Rich Text Format (RTF)																				
4	Unformatted Text																				
5	Metafile																				
6	Device-Independent Bitmap																				
7	Bitmap																				
8	OLE Client Link																				
<b>ABC Equivalent</b>	The <b>PasteSpecial</b> method is equivalent to choosing Paste Special in the ABC Edit menu and then specifying the format to use for the paste. Specifying that the object on the Clipboard be pasted as an icon is equivalent to selecting the Display As Icon option in the Paste Special dialog box.																				

## Related Topics

### Language Elements

[ClipboardFormatAvailable Property](#)

[Copy Method](#)

[Cut Method](#)

[Duplicate Method \(Chart Object\)](#)

[Paste Method](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Using Special Clipboard Formats](#)

### Object

[Chart Object](#)

## PasteLink Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.PasteLink</i> [ <i>HorizontalLocation</i> ] [, <i>VerticalLocation</i> ] The optional <i>HorizontalLocation</i> element is the horizontal location of the paste. The optional <i>VerticalLocation</i> element is the vertical location of the paste.
<b>Description</b>	You use the <b>PasteLink</b> method to paste the contents of the Clipboard into the chart and link the file that is the source of the contents of the chart. You can optionally specify a horizontal and vertical location for the paste. You set the units used for the location using the <b>Units</b> property.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the paste link was successful; False means the paste link was not successful.
<b>ABC Equivalent</b>	The <b>PasteLink</b> method is equivalent to choosing Paste Link in the ABC Edit menu.

## Related Topics

### Language Elements

[DoVerb Method](#)

[InsertObjectFromFile Method](#)

[ObjectType Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[UpdateFields Method](#)

### Description

[Using OLE Client Objects](#)

### Object

[Chart Object](#)

## Related

### PasteLink Method Example

This example uses the **PasteLink** method of the Chart object to paste link an object on the Clipboard into a chart. For the paste to work, there must be something with an OLE Link format available in the Clipboard. For example, you can put an appropriate item in the Clipboard by opening a .BMP file in Paintbrush, selecting a section of it using the dotted rectangle tool, and choosing Copy in the Edit menu.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

If Chart.ClipboardFormatAvailable(8) Then
    Chart.PasteLink 2, 2
End If
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Is OLE Link available in Clipboard?  
' Paste Link at chart coordinates

## PrintOut Method Related

Related

Related

### Usage

*ChartObject*.**PrintOut** [*FromPage*] [, *ToPage*] [, *Copies*] [, *FitToPage*] [, *PrintNotes*]

The *FromPage* element, which is optional, specifies the starting page. The default is the first page.

The *ToPage* element specifies the ending page. The default is the last page.

The *Copies* element, which is optional, specifies the number of copies. The default is 1.

The *FitToPage* element, which is optional, specifies whether to fit the entire chart to one page. The default is False.

The *PrintNotes* element, which is optional, specifies whether to print notes attached to the chart. The default is False.

### Description

You use the **PrintOut** method to print the Chart object.

### Data Type

Integer (Boolean)

### Value

True means that the chart was printed successfully; False means that the chart did not print successfully.

The elements in the **PrintOut** method indicate the options to use when printing.

Element	Description
---------	-------------

<i>FromPage</i>	Integer (default is page 1)
-----------------	-----------------------------

<i>ToPage</i>	Integer (default is last page)
---------------	--------------------------------

<i>Copies</i>	Integer (default is 1)
---------------	------------------------

<i>FitToPage</i>	Integer (Boolean) (default is False)
------------------	--------------------------------------

<i>PrintNotes</i>	Integer (Boolean) (default is False)
-------------------	--------------------------------------

### ABC Equivalent

The **PrintOut** method is equivalent to choosing the Print command in the File menu and choosing printing options.

## **Related Topics**

### **Language Elements**

[PrintSelected Method](#)

[Printer Property](#)

### **Description**

[Printing Charts](#)

[Printing Notes](#)

### **Object**

[Chart Object](#)

## Related

### PrintOut Method Example

This example uses the **PrintOut** method of the Chart object to print a chart from page 1 through page 2, with two copies. The chart is not made to fit to the page and notes are not printed.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True

Set Chart = ABC.Open("c:\abc\tutorial\view.af3")

Chart.PrintOut 1, 2, 2, 0, 0
```

' Start ABC  
' Make ABC visible  
' Open a chart  
' Print chart with these parameters

## PrintSelected Method

Related

Related

Related

### Usage

*ChartObject*.**PrintSelected** [*Copies*] [, *FitToPage*] [, *PrintNotes*]

The *Copies* element, which is optional, specifies the number of copies. The default is 1.  
The *FitToPage* element, which is optional, specifies whether to fit the entire chart to one page. The default is False.

The *PrintNotes* element, which is optional, specifies whether to print notes attached to the chart. The default is False.

### Description

You use the **PrintSelected** method to print the selected objects in the chart.

### Data Type

Integer (Boolean)

### Value

True means that the chart was printed successfully; False means that the chart did not print successfully.

The elements in the **PrintSelected** method indicate the options to use when printing.

#### Element Description

*Copies* Integer (default is 1)

*FitToPage* Integer (Boolean) (default is False)

*PrintNotes* Integer (Boolean) (default is False)

### ABC Equivalent

The **PrintSelected** method is equivalent to choosing the Print command in the File menu and choosing the Print Range Selected option.

## **Related Topics**

### **Language Elements**

[PrintOut Method](#)

[Printer Property](#)

### **Description**

[Printing Charts](#)

[Printing Notes](#)

### **Object**

[Chart Object](#)



## RevertToSaved Method

Related

Related

Related

**Usage** *ChartObject.RevertToSaved*

**Description** Use the **RevertToSaved** method to revert to the last saved copy of the document, discarding any changes.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Save Method](#)

### Description

[Reverting to the Last Saved Version](#)

### Object

[Chart Object](#)

**Related**

## RevertToSaved Method Example

This example uses the **RevertToSaved** method of the Chart object to revert to the saved version of a chart.

```
Dim ABC As Object, Chart As Object
Const MB_YesNo = 4, IDYes = 6, IDNo = 7

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

If Chart.HasDiskFile Then                               ' Has the chart been saved to disk?
    If MsgBox("Chart is saved. Revert to last saved?", MB_YesNo; "Revert to Saved") = IDYes Then
        Chart.RevertToSaved                             ' Revert to last saved copy of chart
    End If
End If
```

## Save Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> . <b>Save</b> [ <i>Path</i> ] [, <i>FileType</i> ] The <i>Path</i> element, which is optional, is a full or partial pathname and filename for the save. The <i>FileType</i> element, which is optional, specifies the type of file to save.										
<b>Description</b>	You use the <b>Save</b> method to save the current chart to disk. If the chart name ends in .AF2 and you do not specify a name, the chart is saved to a new file with an .AF3 extension. If you specify a partial pathname and filename in the optional first element, the value of DefaultFilePath determines the path. You use the second element to specify whether to save the file as a chart or a template, and whether to save as a version 3.0 or 2.0 file. The default is to save the file as a version 3.0 chart.										
<b>Data Type</b>	Integer (Boolean)										
<b>Value</b>	True means that the chart was saved successfully; False means that the chart was not saved successfully. The following table shows the possible values for the second element and their meanings.										
	<table><thead><tr><th><b>FileType</b></th><th><b>Save File As</b></th></tr></thead><tbody><tr><td>0</td><td>Chart, version 3.0</td></tr><tr><td>1</td><td>Template, version 3.0</td></tr><tr><td>2</td><td>Chart, version 2.0</td></tr><tr><td>3</td><td>Template, version 2.0</td></tr></tbody></table>	<b>FileType</b>	<b>Save File As</b>	0	Chart, version 3.0	1	Template, version 3.0	2	Chart, version 2.0	3	Template, version 2.0
<b>FileType</b>	<b>Save File As</b>										
0	Chart, version 3.0										
1	Template, version 3.0										
2	Chart, version 2.0										
3	Template, version 2.0										
<b>ABC Equivalent</b>	The <b>Save</b> method is equivalent to choosing Save in the File menu and specifying a path, name, and type for the file.										

## Related Topics

### Language Elements

[HasDiskFile Property](#)

[Saved Property](#)

### Description

[Saving Charts](#)

### Object

[Chart Object](#)

**Related**

## Save Method, CloseChart Method, and Saved Property Example

This example uses the **Saved** property, **Save** method, and **CloseChart** method of the Chart object to find out if a chart is saved, save it, and close it. Also see the example included with the [HasDiskFile property of the Chart object](#).

```
Dim ABC As Object, Chart As Object
Dim Obj1 As Object, Obj2 As Object, Line1 As Object
Const MB_YesNo = 4, IDYes = 6, IDNo = 7

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Obj1 = Chart.DrawShape("Operation")           ' Draw a shape
Obj1.Text = "Unit 1"                              ' Add text to the shape
Set Obj2 = Chart.DrawShape("Decision")           ' Draw a shape
Obj2.Text = "Unit 2"                              ' Add text to the shape

Set Line1 = Chart.DrawLine(Obj1, Obj2, 0, 2)      ' Draw a line between two shapes

Chart.Repaint                                     ' May be needed for some video modes

If Not Chart.Saved Then                           ' Is this chart saved?
    Chart.Save "C:\ABC\TUTORIAL\TEST2.AF3"        ' Save the chart as TEST2
End If

If MsgBox("Chart is saved. Ready to close?", MB_YesNo; "ABC FlowCharter") = IDYes Then
    Chart.CloseChart                              ' Close the chart
End If
```

## ScrollPage Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.ScrollPage PageNumber</i> The <i>PageNumber</i> element is the page to which to scroll.
<b>Description</b>	You use the <b>ScrollPage</b> method to scroll the chart to a particular page.
<b>Data Type</b>	None
<b>Value</b>	The page to which to scroll
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[PageCount Property](#)  
[ScrollLeft Property](#)  
[ScrollPosition Method](#)  
[ScrollTop Property](#)  
[View Property](#)

### Description

[Viewing a Chart](#)

### Object

[Chart Object](#)

**Related**

## ScrollPage Method Example

This example uses the **ScrollPage** method of the Chart object to scroll to the last page of a chart.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

For X = 1 To 10                                        ' Draw shapes
    Set Shape = Chart.DrawShape("Decision")
Next X

Chart.ScrollPage (Chart.PageCount)                    ' Scroll to the last page of chart
```

## ScrollPosition Method

Related

Related

Related

### Usage

*ChartObject*.**ScrollPosition** *LeftDistance*, *TopDistance*

The *LeftDistance* element specifies the left part of the chart area to show in the window.

The *TopDistance* element specifies the top part of the chart area to show in the window.

### Description

You use the **ScrollPosition** method to scroll to a location in the chart.

### Value

The *LeftDistance* and *TopDistance* elements are each doubles.

**ABC Equivalent** None

## Related Topics

### Language Elements

[PageCount Property](#)

[ScrollLeft Property](#)

[ScrollPage Method](#)

[ScrollTop Property](#)

[View Property](#)

### Description

[Viewing a Chart](#)

### Object

[Chart Object](#)



## Select Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.Select (Value)</i> The <i>Value</i> element specifies what to select.										
<b>Description</b>	You use the <b>Select</b> method to select specified types of objects or to deselect all objects. The <b>Select</b> method with a value of 3 has the same effect as the <b>DeselectAll</b> method.										
<b>Value</b>	The values for the types are in the following table. <table><thead><tr><th><b>Value</b></th><th><b>Action</b></th></tr></thead><tbody><tr><td>0</td><td>Selects all shapes</td></tr><tr><td>1</td><td>Selects all lines</td></tr><tr><td>2</td><td>Selects everything</td></tr><tr><td>3</td><td>Deselects everything</td></tr></tbody></table>	<b>Value</b>	<b>Action</b>	0	Selects all shapes	1	Selects all lines	2	Selects everything	3	Deselects everything
<b>Value</b>	<b>Action</b>										
0	Selects all shapes										
1	Selects all lines										
2	Selects everything										
3	Deselects everything										
<b>ABC Equivalent</b>	The <b>Select</b> method is equivalent to choosing Select in the ABC Edit menu and choosing Shapes (0), Lines (1), or All (2).										

## Related Topics

### Language Elements

[DeselectAll Method](#)

[Selected Property](#)

[SelectShape Type Method](#)

### Description

[Selecting Objects in a Chart](#)

[Selecting Shapes](#)

### Object

[Chart Object](#)

## Related

# Select Method, Copy Method, Paste Method, and ClipboardFormatAvailable Property Example

This example uses the **Select** method, **Copy** method, **ClipboardFormatAvailable** property, and **Paste** method of the Chart object to select a shape, copy it to the Clipboard, check the type of data in the Clipboard, and paste from the Clipboard.

```
Dim ABC As Object, Chart As Object, Obj1 As Object
Dim PasteIt

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Obj1 = Chart.DrawShape("Operation")
Obj1.Text = "Unit 1"

Chart.Select (0)
Chart.Copy

If Chart.ClipboardFormatAvailable(0) Then
    PasteIt = Chart.Paste(2, 2)
End If
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' Draw Operation shape
' Add text to shape

' Select shapes
' Copy shape to Clipboard

' Is Clipboard ABC Native data?
' Paste shape
```

## SelectShapeType Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.SelectShapeType ShapeName</i> The <i>ShapeName</i> element is a string that specifies the type of shapes to select.
<b>Description</b>	You use the <b>SelectShapeType</b> method to select all shapes of a specific type, such as Decision (diamond) shapes.
<b>Value</b>	The <i>ShapeName</i> element is a string that specifies the type of shapes you want to select.
<b>ABC Equivalent</b>	None

## **Related Topics**

### **Language Elements**

[DeselectAll Method](#)

[Select Method](#)

[Selected Property](#)

### **Description**

[Selecting Objects in a Chart](#)

[Selecting Shapes](#)

### **Object**

[Chart Object](#)

**Related**

## SelectShapeType Method Example

This example uses the **SelectShapeType** method of the Chart object to select shapes of a particular type.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.DrawSpacingX = 1.5                               ' Set horizontal spacing for shapes

For X = 1 To 3                                         ' Draw shapes
    Set Obj1 = Chart.DrawShape("Document")
    Set Obj2 = Chart.DrawShape("Decision")
Next X

Chart.SelectShapeType "Decision"                       ' Select all Decision shapes
```

## SetProtection Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.SetProtection Switch, Password</i> The <i>Switch</i> element specifies whether protection is on or off. The <i>Password</i> element specifies a password for the chart.
<b>Description</b>	You use the <b>SetProtection</b> method to turn protection on and off by setting a protection value and a password.
<b>Data Type</b>	None
<b>Value</b>	The <i>Switch</i> element is an integer (Boolean). True means the chart is protected; False means the chart is not protected. The <i>Password</i> element is a string specifying the password for the chart.
<b>ABC Equivalent</b>	The <b>SetProtection</b> method is equivalent to choosing Protect Chart in the File menu and entering a password.

## Related Topics

**Language Elements**

[Protected Property](#)

**Description**

[Protecting Charts](#)

**Object**

[Chart Object](#)

## Related

### SetProtection Method Example

This example uses the **SetProtection** method of the Chart object to set a password for a chart. After the program runs, you must choose Unprotect Chart in the File menu and type the password "turtle" to unprotect the chart.

```
Dim ABC As Object, Chart As Object, Shape As Object
```

```
Set ABC = CreateObject("ABCFlow.application")  
ABC.Visible = True  
Set Chart = ABC.ActiveChart
```

```
' Start ABC  
' Make ABC visible  
' Get the active chart
```

```
Chart.SetProtection 1,"turtle"
```

```
' Set password
```

## Spelling Method

Related

Related

Related

**Usage** *ChartObject.Spelling*

**Description** The **Spelling** method lets you start spell checking the chart.

**ABC Equivalent** The **Spelling** method is equivalent to choosing the text you want to check and choosing Spelling in the Edit menu.

## Related Topics

### Description

[Checking Spelling](#)

### Object

[Chart Object](#)



## ToBack Method (Chart Object)

Related

Related

Related

### Usage

*ChartObject.ToBack*

### Description

You use the **ToBack** method of the Chart object to move all selected objects in the chart to the back.

### ABC Equivalent

The **ToBack** method is equivalent to selecting objects and clicking the To Back button in the ABC Selection ribbon.

## Related Topics

### Language Elements

[ToBack Method \(Object Object\)](#)

[ToFront Method \(Chart Object\)](#)

[ToFront Method \(Object Object\)](#)

### Description

[Changing the Display Order of Objects](#)

### Object

[Chart Object](#)



## ToFront Method (Chart Object)

Related

Related

Related

**Usage** *ChartObject.ToFront*

**Description** You use the **ToFront** method of the Chart object to move all selected objects to the front.

**ABC Equivalent** The **ToFront** method is equivalent to selecting objects and clicking the To Front button in the ABC Selection ribbon.

## **Related Topics**

### **Language Elements**

[ToBack Method \(Chart Object\)](#)

[ToBack Method \(Object Object\)](#)

[ToFront Method \(Object Object\)](#)

### **Description**

[Changing the Display Order of Objects](#)

### **Object**

[Chart Object](#)

## UpdateFields Method

Related

Related

Related

**Usage** *ChartObject.UpdateFields*

**Description** The **UpdateFields** method updates all the fields for all the linked shapes in a chart so they reflect the values in the linked charts.

**ABC Equivalent** The **UpdateFields** method is equivalent to choosing Update in the Fields menu.

## Related Topics

### Language Elements

[DoVerb Method](#)

[InsertObjectFromFile Method](#)

[IsLinked Property](#)

[LinkedChartName Property](#)

[LinkFields Property](#)

[LinkIndicator Property](#)

[LinkNOTIFY Event](#)

[LinkShadow Property](#)

[ObjectType Property](#)

[PasteLink Method](#)

### Description

[Using OLE Client Objects](#)

[Using Linked Field Data](#)

### Object

[Chart Object](#)



## Restore Method (Chart Object)

Related

Related

Related

**Usage** *ChartObject.Restore*

**Description** The **Restore** method of the Chart object lets you change the chart window to its previous size

**ABC Equivalent** The **Restore** method is equivalent to clicking the restore arrow in the upper right of the chart window.

## Related Topics

### Language Elements

[Maximize Method \(Chart Object\)](#)

[Minimize Method \(Chart Object\)](#)

[Restore Method \(Application Object\)](#)

### Description

[Minimizing, Maximizing, and Restoring a Window](#)

### Object

[Chart Object](#)

## Minimize Method (Chart Object)

Related

Related

Related

### Usage

*ChartObject.Minimize*

### Description

The **Minimize** method of the Chart object lets you change a chart window to an icon.

### ABC Equivalent

The **Minimize** method is equivalent to clicking the chart minimize arrow in the upper right of the chart window.

## **Related Topics**

### **Language Elements**

[Maximize Method \(Chart Object\)](#)

[Minimize Method \(Application Object\)](#)

[Restore Method \(Chart Object\)](#)

### **Description**

[Minimizing, Maximizing, and Restoring a Window](#)

### **Object**

[Chart Object](#)

## Maximize Method (Chart Object)

Related

Related

Related

**Usage** *ChartObject.Maximize*

**Description** The **Maximize** method of the Chart object lets you change a chart window to its maximum size.

**ABC Equivalent** The **Maximize** method is equivalent to clicking the chart maximize arrow in the upper right of the chart window.

## Related Topics

### Language Elements

[Maximize Method \(Application Object\)](#)

[Minimize Method \(Chart Object\)](#)

[Restore Method \(Chart Object\)](#)

### Description

[Minimizing, Maximizing, and Restoring a Window](#)

### Object

[Chart Object](#)



## NextShapeHeight Property

Related

Related

Related

**Usage** *ChartObject.NextShapeHeight = Height*

**Description** The **NextShapeHeight** property lets you find or set the height of the next shape to be drawn.

**Data Type** Double

**Value** The height of the next shape to be drawn

**ABC Equivalent** None

## Related Topics

### Language Elements

[NextShapeWidth Property](#)

### Description

[Drawing Shapes](#)

### Object

[Chart Object](#)

## Related

### NextShapeHeight, NextShapeWidth Properties Example

This example uses the **NextShapeHeight** property and **NextShapeWidth** property of the Chart object to set the height and width of the next shape drawn.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim X

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

For X = 1 To 3
    Set NewShape = Chart.DrawShape("Operation")         ' Draw a shape
    Chart.NextShapeHeight = Chart.NextShapeHeight + .25 ' Set height of next shape
    Chart.NextShapeWidth = Chart.NextShapeWidth + .25  ' Set width of next shape drawn
Next X
```

## NextShapeWidth Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.NextShapeWidth = Width</i>
<b>Description</b>	The <b>NextShapeWidth</b> property lets you find or set the width of the next shape to be drawn.
<b>Data Type</b>	Double
<b>Value</b>	The width of the next shape to be drawn
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[NextShapeHeight Property](#)

### Description

[Drawing Shapes](#)

### Object

[Chart Object](#)

## GroupAndLink Method

Related

Related

Related

### Usage

*ChartObject*.**GroupAndLink** ([*NewChartPath*] [, *FieldsLinked*])

The *NewChartPath* element specifies the full pathname of the new chart. If you omit the element, ABC generates a default chart pathname.

The *FieldsLinked* element specifies whether the new chart's fields are linked to the source chart. If you omit the second element, ABC does not link the fields.

### Description

The **GroupAndLink** method lets you move a group of selected objects to another chart and replace the moved group with a shape that is linked to the chart to which the group was moved. The **GroupAndLink** method returns the shape that replaced the moved group and has two optional elements. After executing **GroupAndLink**, you can obtain the newly created chart object with the **ActiveChart** property of the Application object.

### Data Type

Object

### Value

The chart that is created. The *NewChartPath* element is a string. The *FieldsLinked* element is a Boolean.

### ABC Equivalent

The **GroupAndLink** method is equivalent to choosing two or more shapes, clicking the Link button in the Select ribbon, and then selecting Group And Link in the Link dialog box.

## Related Topics

### Language Elements

[ActiveChart Property](#)

[Link Method](#)

[LinkIndicator Property](#)

[LinkShadow Property](#)

### Description

[Creating Group Links](#)

### Object

[Chart Object](#)

## Related

### GroupAndLink Method Example

This example uses the **GroupAndLink** method of the Chart object to group and link selected shapes to another file.

```
Dim ABC As Object, Chart As Object, Obj1 As Object, Obj2 As Object
Dim X As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
SamplePath = ABC.Path + "\SAMPLES\QUALITY.AF3"

Set Chart = ABC.ActiveChart                            ' Set chart object
Set Obj1 = Chart.DrawShape("Operation")                ' Draw operation shape
Obj1.Text = "(Linked) Shape 1"                         ' Add text to shape

Chart.Select (0)                                       ' Select shape
Chart.GroupAndLink (SamplePath)                       ' Group and link selected shapes
chart

Set Obj2 = Chart.DrawShape("Decision")                 ' Draw shape decision shape
Obj2.Text = "(Non Linked) Shape 2"                     ' Add text to shape
```

## TypeRequiresEXE Property

Related

Related

Related

### Usage

*ChartObject.TypeRequiresEXE* = {True | False}

### Description

You can link a compiled Visual Basic EXE program file to a chart so that the EXE program runs automatically when you open the chart. If you set the **TypeRequiresEXE** property to True, the chart requires the EXE to open. If the linked EXE cannot be run, then the chart does not open. The name of the EXE that is linked to a chart is constructed by adding .EXE to the chart Type. **Note:** ABC only runs one instance of a linked EXE. When a second chart that is linked to an already running EXE is loaded, ABC refers to the currently running EXE. It does not load a second copy of the EXE. If you set either **TypeRequiresEXE** or **TypeUsesEXE** to True in a program, then you must also ensure that you close all charts of that Type when your program closes. You can use the **ChartTypeShutdown** method to close the charts.

### Data Type

Integer (Boolean)

### Value

True means the chart type always runs an associated program (EXE); False means it does not.

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[ChartTypeShutdown Method](#)

[Type Property \(Chart Object\)](#)

[TypeUsesEXE Property](#)

### **Description**

[Linking EXEs to Charts](#)

### **Object**

[Chart Object](#)

## Related

### TypeRequiresEXE, TypeUsesEXE Properties Example

This example uses the **TypeRequiresEXE** property and the **TypeUsesEXE** property of the Chart object to require or permit use of an executable program.

The following code is placed in the declarations section.

```
Const CHARTTYPE = "PROJECT1"  
Const APPNAME = "Form1"
```

The following code is placed in the form.

```
Dim ABC As Object, Chart As Object  
Dim ChartUnits As Integer  
  
Set ABC = CreateObject("ABCFlow.application")           ' Start ABC  
ABC.Visible = True                                     ' Make ABC visible  
  
Set Chart = ABC.ActiveChart  
Chart.DrawShape ("Decision")                          ' Draw a decision shape  
Chart.DrawTextBlock "This chart requires that 'Form1' be loaded when this chart is opened."  
Chart.Type = CHARTTYPE                                ' Sets the Chart Type to the constant  
' Require that "PROJECT1" be loaded when this chart is opened  
Chart.TypeRequiresEXE = True  
' Load the CHARTTYPE when charts created with this application are opened  
Chart.TypeUsesEXE = True  
MsgBox "Save this chart. Close ABC. Switch to Visual Basic and stop the running application. Make  
the stopped application an .EXE. Close Visual Basic and load ABC FlowCharter. Open the chart you  
saved."
```

The following code is placed in the QueryUnload section of the form.

```
x = ABC.ChartTypeShutdown(CHARTTYPE, APPNAME)
```

## TypeUsesEXE Property

Related

Related

Related

### Usage

*ChartObject.TypeUsesEXE* = {True | False}

### Description

You can link a compiled Visual Basic EXE program file to a chart so that the EXE program attempts to run when you open the chart. If you set the **TypeUsesEXE** property to True, then the chart attempts to run the linked EXE when it opens. If the EXE cannot be run, the chart still opens. The name of the EXE that is linked to a chart is constructed by adding .EXE to the chart Type. **Note:** ABC only runs one instance of a linked EXE. When a second chart that is linked to an already running EXE is loaded, ABC refers to the currently running EXE. It does not load a second copy of the EXE. If you set either **TypeRequiresEXE** or **TypeUsesEXE** to True in a program, then you must also ensure that you close all charts of that Type when your program closes. You can use the **ChartTypeShutdown** method to close the charts.

### Data Type

Integer (Boolean)

### Value

True means the chart type attempts to runs an associated program (EXE); False means it does not.

**ABC Equivalent** None

## Related Topics

### Language Elements

[ChartTypeShutdown Method](#)

[Type Property \(Chart Object\)](#)

[TypeRequiresEXE Property](#)

### Description

[Linking EXEs to Charts](#)

### Object

[Chart Object](#)

## LineCrossoverSize Property

Related

Related

Related

### Usage

*ChartObject.LineCrossoverSize = RelativeSize*

### Description

The **LineCrossoverSize** property lets you find or set the size of the crossover when one line crosses of another. The setting applies to bunny hops and broken lines, but has no effect when the crossover style is solid lines. (See the [LineCrossoverStyle property](#) for information on the available styles.) The **LineCrossoverSize** property is read/write.

### Data Type

Integer

### Value

The values for the relative sizes for bunny hop crossovers are in the following table. The same relative sizes apply when the style is broken lines.

#### **RelativeSize**   **Description**

0   **Related**   Small

1   **Related**   Medium

2   **Related**   Large

### Value

The relative size of the crossover when one line crosses another

### ABC Equivalent

The **LineCrossoverSize** property is equivalent to choosing Preferences in the ABC File menu, clicking the Line Options button, and dragging the slider in the Crossovers section to set the size of the crossover.

## Related Topics

### Language Elements

[LineCrossoverStyle Property](#)

[ShowNodesOnLines](#)

### Description

[Line Options](#)

[Setting Line Crossovers](#)

### Object

[Chart Object](#)

## Related

# LineCrossoverStyle Property and LineCrossoverSize Property Example

This example uses the **LineCrossoverStyle** property and **LineCrossoverSize** property of the Chart object to set the style and size used when one line crosses another.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True ' Make ABC visible
Set Chart = ABC.New ' Make a new chart

Chart.LineCrossoverStyle = 1 ' Set style to broken lines
Chart.LineCrossoverSize = 2 ' Set size to large
Chart.MasterItems.HideAll ' Get Master Items out of the way

Chart.DrawPositionX = 1 ' Draw 2 shapes and connect...
Chart.DrawPositionY = 2 ' ... them with a line
Set Shape1 = Chart.DrawShape
Chart.DrawPositionX = 4
Set Shape2 = Chart.DrawShape
Chart.DrawLine Shape1, Shape2

Chart.DrawPositionX = 1 ' Draw 2 more shapes and connect...
Chart.DrawPositionY = 1 ' ... them with an overlapping line
Set Shape1 = Chart.DrawShape
Chart.DrawPositionX = 3
Chart.DrawPositionY = 3
Set Shape2 = Chart.DrawShape
Chart.DrawLine Shape1, Shape2
```

## LineCrossoverStyle Property

Related

Related

Related

### Usage

*ChartObject.LineCrossoverStyle = Style*

### Description

The **LineCrossoverStyle** property lets you find or set the style of the crossover when one line crosses another. The **LineCrossoverStyle** property is read/write.

### Data Type

Integer

### Value

The values for the styles are in the following table.

Style	Description
-------	-------------

0	Related Bunny hops
---	--------------------

1	Related Broken lines
---	----------------------

2	Related Solid lines
---	---------------------

### Value

The style when one line crosses another

### ABC Equivalent

The **LineCrossoverStyle** property is equivalent to choosing Preferences in the ABC File menu, clicking the Line Options button, and choosing a crossover style.

## Related Topics

### Language Elements

[LineCrossoverSize Property](#)

[ShowNodesOnLines](#)

### Description

[Line Options](#)

[Setting Line Crossovers](#)

### Object

[Chart Object](#)

## ZoomPercentage Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.ZoomPercentage = Percentage</i>
<b>Description</b>	The <b>ZoomPercentage</b> property lets you find or set the view of the current chart as a percentage of actual size. The <b>ZoomPercentage</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The view of the current chart as a percentage of actual size. The ZoomPercentage property can be from 25 to 400 (25% to 400%).
<b>ABC Equivalent</b>	The <b>ZoomPercentage</b> property is equivalent to choosing the View tool, choosing the View Zoom Percentage button, entering the percent view you want in the Choose Zoom Percentage dialog box, and clicking OK.

## Related Topics

Language Elements

[View](#)

Description

[Viewing a Chart](#)

Object

[Chart Object](#)

Related

## ZoomPercentage Properties Example

This example uses the **ZoomPercentage** property of the Chart object to change the view of the current chart.

```
Dim ABC As Object, Chart As Object
```

```
Set ABC = CreateObject("ABCFlow.application")  
ABC.Visible = True  
Set Chart = ABC.ActiveChart  
Chart.ChartZoomPercentage = 50
```

```
' Start ABC  
' Make ABC visible  
' Get the active chart  
' Set zoom to 50%
```

## SendMail Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> . <b>SendMail</b>
<b>Description</b>	The <b>SendMail</b> method creates a new e-mail message with the chart object as an attachment. The user addresses the e-mail and creates a message as he or she usually does. The <b>SendMail</b> method uses the MAPI e-mail system and is compatible with Microsoft Mail.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the e-mail message was created successfully; False means the creation was not successful.
<b>ABC Equivalent</b>	The <b>SendMail</b> method is equivalent to choosing Send Mail in the ABC File menu.

## Related Topics

### Description

[Sending Electronic Mail](#)

### Object

[Chart Object](#)

**Related**

## SendMail Method Example

This example uses the **SendMail** method of the Chart object to launch a new e-mail message with the chart attached. The user must address the e-mail and create a message as he or she usually does.

```
Dim ABC As Object, Chart As Object
```

```
Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True ' Make ABC visible
Set Chart = ABC.ActiveChart ' Get the active chart
Chart.SendMail ' Launches a new e-mail with this chart attached
```

## SetDefaults Method

Related

Related

Related

### Usage

*ChartObject*.**SetDefaults** *DefaultObject*

The *DefaultObject* element is an object that has the properties that you want to be the new defaults.

### Description

The **SetDefaults** method sets the defaults for subsequent objects. The *DefaultObject* is a Shape, Line\_, or TextBlock object. Subsequent objects of that type have the defaults you set.

The following table lists the defaults that you set when you use the **SetDefaults** method.

#### Object Type Defaults Set

Shape	Border color, border style, border width, fill color, fill pattern, shadow offset, shadow color, numbers on or off, font properties, text alignment
Line_	Color, width, style, source arrow size, source arrow style, source arrow color, destination arrow size, destination arrow style, destination arrow color
TextBlock	Font properties, text alignment

### Data Type

Integer (Boolean)

### Value

True means the defaults were created successfully; False means the defaults were not created successfully.

**ABC Equivalent** None

## Related Topics

### Description

[Setting Defaults](#)  
[Speeding Actions](#)

### Object

[Chart Object](#)

## Related

### SetDefaults, Clear\_ Methods Example

This example uses the **SetDefaults** method of the Chart object and the **Clear\_** method of the Object object to set the defaults for shapes using a dummy object and then delete the object.

```
Dim ABC As Object
Dim Chart As Object
Dim Obj As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

    Set Obj = Chart.DrawShape

Obj.Shape.NumberShown = True
Obj.Shape.FillColor = ABC.GRAY
Obj.Shape.ShadowColor = ABC.DK_Gray
Obj.Shape.ShadowStyle = 1

Chart.SetDefaults Obj.Shape
Obj.Clear_

' Start ABC
' Make ABC visible
' Get the active chart

' Create a dummy shape to hold the defaults

' Turn shape numbering on and set...
' ...all newly drawn shapes to gray...
' ...with dark gray shadows

' Set the defaults for newly draw shapes
' Clear the dummy Object
```

## FullScreen Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject</i> . <b>FullScreen</b>
<b>Description</b>	The <b>FullScreen</b> method shows the chart on the full screen without menus or buttons. Use the <b>CancelFullScreen</b> method to return to the previous view.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the chart was shown successfully; False means the chart was not shown successfully.
<b>ABC Equivalent</b>	The <b>FullScreen</b> method is equivalent to choosing Full Screen in the ABC Window menu or the ABC View menu.

## Related Topics

### Language Elements

[CancelFullScreen Method](#)

### Description

[Giving a presentation](#)

### Object

[Chart Object](#)



## CancelFullScreen Method

Related

Related

Related

<b>Usage</b>	<i>ChartObject.CancelFullScreen</i>
<b>Description</b>	The <b>CancelFullScreen</b> method returns a chart to its previous view after you have used the <b>FullScreen</b> method to show it on the full screen without menus or buttons.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the chart was shown successfully; False means the chart was not shown successfully.
<b>ABC Equivalent</b>	The <b>CancelFullScreen</b> method is equivalent to pressing <b>Esc</b> to leave the full screen view.

## Related Topics

### Language Elements

[FullScreen Method](#)

### Description

[Giving a presentation](#)

### Object

[Chart Object](#)

## Charts Collection

**Description** The Charts collection is below the Application object. Below the Charts collection are the Chart objects. You can have multiple Chart objects in the Chart collection.

### Properties

---

[Application](#)

[Count](#)

[Parent](#)

### Methods

---

[Add](#)

[AddFromTemplate](#)

[CloseAll](#)

[Item](#)

[Open](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Add Method (Charts Collection)

Related

Related

Related

**Usage** *ChartsCollection.Add*

**Description** You use the **Add** method of the Charts collection to create a new chart with default attributes and automatically add it to the collection.

**Data Type** Object

**Value** The chart that is created

**ABC Equivalent** The **Add** method is equivalent to choosing the New command in the File menu.

## Related Topics

### Language Elements

[Add method \(FieldTemplates Collection\)](#)

[AddFromTemplate Method](#)

[New Method](#)

[NewFromTemplate Method](#)

### Description

[Creating New Charts](#)

### Object

[Charts Collection](#)



## AddFromTemplate Method

Related

Related

Related

<b>Usage</b>	<i>ChartsCollection.AddFromTemplate (TemplateName)</i> The <i>TemplateName</i> element is the path and name of the template to use to create the chart.
<b>Description</b>	You use the <b>AddFromTemplate</b> method to create a new chart based on the specified chart template name. If <i>TemplateName</i> file cannot be loaded for any reason, the returned <i>Chart.Valid</i> is False.
<b>Data Type</b>	Object
<b>Value</b>	The chart that is created
<b>ABC Equivalent</b>	The <b>AddFromTemplate</b> method is equivalent to choosing the New From Template command in the File menu.

## Related Topics

### Language Elements

[Add Method \(Charts Collection\)](#)

[New Method](#)

[NewFromTemplate Method](#)

### Description

[Creating New Charts](#)

### Object

[Charts Collection](#)

## Item Method (Chart Objects Collection)

Related

Related

Related

<b>Usage</b>	<i>ChartsCollection.Item</i> ( <i>{PathName   Number}</i> ) The <i>PathName</i> element is a string indicating the full path and executable name of the chart. The <i>Number</i> element is the chart's ordering position within the collection.
<b>Description</b>	The <b>Item</b> method of the Charts collection lets you retrieve a chart from the Charts collection. The method returns a nonvalid chart object if the specified chart object does not exist.
<b>Data Type</b>	Object
<b>Value</b>	The Chart object
<b>ABC Equivalent</b>	The <b>Item</b> method of the Charts collection is equivalent to opening the Window menu and choosing the chart from the numbered list of open charts.

## Related Topics

### Language Elements

- [Activate Method \(Chart Object\)](#)
- [ActiveChart Property](#)
- [Application Property](#)
- [Count Property](#)
- [Item Method \(FieldTemplates Collection\)](#)
- [Item Method \(FieldValues Collection\)](#)
- [Item Method \(Menu Collection\)](#)
- [Item Method \(Objects Collection\)](#)
- [Name Property \(Chart Object\)](#)

### Description

- [Activating a Chart](#)

### Object

- [Charts Collection](#)



## FieldTemplate Object

**Description** The FieldTemplate object is below the FieldTemplates collection. You can have multiple FieldTemplate objects.

### **Properties**

---

[Accumulation](#)  
[AccumulationMethod](#)  
[Application](#)  
[Format](#)  
[Hidden](#)  
[Name](#)  
[Parent](#)  
[Type](#)

### **Methods**

---

There are no methods for the FieldTemplate object.

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Accumulation Property

Related

Related

Related

<b>Usage</b>	<i>FieldTemplateObject.Accumulation</i>
<b>Description</b>	The <b>Accumulation</b> property lets you find the accumulated value of data fields, as the value will appear in the Legend. You set the type of accumulation using the <b>AccumulationMethod</b> property. The <b>Accumulation</b> property is read only.
<b>Data Type</b>	Double
<b>Value</b>	The value of the <b>Accumulation</b> property is the accumulated value of a FieldTemplate object. The Field Template object can be any data field that is added to a chart.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[AccumulationMethod Property](#)

[Item Method \(FieldTemplates Collection\)](#)

[ShowLegend Property](#)

[Value Property](#)

### Description

[Working with Data Field Values](#)

### Object

[FieldTemplate Object](#)

**Related**

## Accumulation Property Example

This example uses the **Accumulation** property of the FieldTemplate object to find a data field's accumulation.

```
Dim ABC As Object, MasterItems As Object, Chart As Object
Dim Field_Template As Object
Dim Field_Accumulation As Double

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory") ' Add a data field

Field_Template.AccumulationMethod = 1                  ' Set accumulation method to sum

Field_Accumulation = Field_Template.Accumulation      ' Get field's accumulation
MsgBox CStr(Field_Accumulation)
```

## AccumulationMethod Property

Related

Related

Related

### Usage

*FieldTemplateObject.AccumulationMethod = Integer*

### Description

The **AccumulationMethod** property lets you find or specify the type of accumulation for a data field. The accumulation is calculated for the Legend. The **AccumulationMethod** property is read/write.

### Data Type

Integer

### Value

The values for the accumulation methods are in the following table.

Value	Accumulation Method
0	No Accumulation: Do not include this field in the Legend.
1	Sum: The total of all field values added together.
2	Mean: The average of all the values.
3	Median: The middle value in the entire range of values.
4	Min: The smallest value in the entire range of values.
5	Max: The largest value in the entire range of values.
6	Range: The difference between the largest and smallest values.
7	Count: The number of values.
8	Nonnull Count: The number of values that are not null.

### ABC Equivalent

The **AccumulationMethod** property is equivalent to opening the ABC Fields menu, choosing an option in the Accumulation Method list box in the Setup Fields dialog box, and clicking OK.

## Related Topics

### Language Elements

- [Accumulation property](#)
- [Add Method \(FieldTemplates Collection\)](#)
- [Format Property](#)
- [Hidden Property](#)
- [Name Property \(FieldValue Object\)](#)
- [ShowLegend Property](#)
- [Type Property \(FieldValue Object\)](#)

### Description

- [Changing Data Field Attributes](#)

### Object

- [ABC Object](#)

**Related**

## AccumulationMethod Property Example

This example uses the **AccumulationMethod** property of the FieldTemplate object to set and find a data field's accumulation method.

```
Dim ABC As Object
Dim MasterItems As Object
Dim Chart As Object
Dim Field_Template As Object
Dim Field_Accumulation_Method As Single

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory") ' Add a data field

Field_Template.AccumulationMethod = 1                  ' Set accumulation method to sum

Field_Accumulation_Method = Field_Template.AccumulationMethod ' Find method

Select Case Field_Accumulation_Method                  ' Report accumulation method
    Case 0
        MsgBox "No Accumulation"
    Case 1
        MsgBox "Sum"
    Case 2
        MsgBox "Mean"
    Case 3
        MsgBox "Median"
    Case 4
        MsgBox "Min"
    Case 5
        MsgBox "Max"
    Case 6
        MsgBox "Range"
    Case 7
        MsgBox "Total"
    Case 8
        MsgBox "Non Null Total"
End Select
```

## Format Property

Related

Related

Related

### Usage

*FieldTemplateObject.Format = Format*

### Description

The **Format** property lets you find or set the format for a data field. The Format property is read/write.

### Data Type

Integer

### Value

The values for the formats of data fields are in the following table.

Value	Duration Format	Value	Date Format
100	# w.	200	M/D/YY
101	# weeks	201	MMMM-D-YY
102	# d.	202	MMMM DD, YYYY
103	# days	203	MMM-YY
104	# h.	204	MMMM YYYY
105	# hrs.		
106	# hours	Value	Currency Format
107	# m.	300	####0.00(####0.00)
108	# mins.	301	#,##0.00(\$,##0.00)
109	# minutes	302	####0(####0)
110	# s.	303	#,##0(\$,##0)
111	# secs.		
112	# seconds	Value	Number Format
113	# TMU	500	###0
114	h:m	501	###0.00
115	m:s	502	###0.0000
116	h:m:s	503	#,##0
		504	#,##0.00
		505	#,##0.0000
Value	Percent Format		
400	##%		
401	#0.00%		

**ABC Equivalent** The **Format** property is equivalent to opening choosing Setup in the ABC Fields menu, selecting a data field, choosing a format for the field in the Setup Fields dialog box, and clicking OK.

## Related Topics

### Language Elements

- [AccumulationMethod Property](#)
- [Add Method \(FieldTemplates Collection\)](#)
- [Hidden Property](#)
- [Name Property \(FieldValue Object\)](#)
- [Type Property \(FieldValue Object\)](#)

### Description

- [Changing Data Field Attributes](#)

### Object

- [FieldTemplate Object](#)

**Related**

## Format Property Example

This example uses the **Format** property of the FieldTemplate object to find a data field's format.

```
Dim ABC As Object
Dim MasterItems As Object, Chart As Object
Dim Field_Template As Object
Dim Field_Accumulation As Double
Dim Field_Accumulation_Method As Single
Dim Field_Format As Single

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory") ' Add a data field

Field_Template.AccumulationMethod = 1                  ' Set accumulation method to sum

Field_Format = Field_Template.Format                   ' Get field format

Select Case Field_Format                               ' Report field format
    Case 100
        MsgBox "# w."
    Case 101
        MsgBox "# weeks"
    Case 102
        MsgBox "# d."
    Case 103
        MsgBox "# days"
    Case 104
        MsgBox "# h."
    Case 105
        MsgBox "# hrs."
    Case 106
        MsgBox "# hours"
    Case 107
        MsgBox "# m."
    Case 108
        MsgBox "# mins."
    Case 109
        MsgBox "# minutes"
    Case 110
        MsgBox "# s."
    Case 111
        MsgBox "# secs."
    Case 112
        MsgBox "# seconds"
    Case 113
        MsgBox "# TMU"
    Case 114
        MsgBox "h:m"
    Case 115
        MsgBox "m:s"
    Case 116
        MsgBox "h:m:s"
    Case 200
        MsgBox "M/d/yy"
    Case 201
        MsgBox "mmm-d-yy"
    Case 202
        MsgBox "MMMMM dd, yyyy"
    Case 203
        MsgBox "mmm-yy"
    Case 204
        MsgBox "MMMMM yy"
    Case 300
        MsgBox "$###0.00 ($###0.00) "
```

```
Case 301
  MsgBox "$#,##0.00($#,##0.00)"
Case 302
  MsgBox "$###0($#,##0.00)"
Case 303
  MsgBox "$#,##0($#,##0)"
Case 400
  MsgBox "##%"
Case 401
  MsgBox "#0.00%"
Case 500
  MsgBox "###0"
Case 501
  MsgBox "###0.00"
Case 502
  MsgBox "###0.0000"
Case 503
  MsgBox "#,##0"
Case 504
  MsgBox "#,##0.00"
Case 505
  MsgBox "#,##0.0000"
End Select
```

## Hidden Property

Related

Related

Related

### Usage

*FieldTemplateObject.Hidden* = {True | False}

### Description

The **Hidden** property lets you find or set whether a data field and its value are displayed in the chart. The **Hidden** property is read/write.

### Data Type

Integer (Boolean)

### Value

True hides a field and its value; False displays a field and its value.

### ABC Equivalent

The **Hidden** property is equivalent to choosing Setup in the ABC Fields menu, selecting a data field, selecting or deselecting the Hidden Field option in the Setup Fields dialog box, and clicking OK.

## Related Topics

### Language Elements

- [Accumulation Property](#)
- [AccumulationMethod Property](#)
- [Add Method \(FieldTemplates Collection\)](#)
- [FieldNamesHidden Property](#)
- [Format Property](#)
- [Name Property \(FieldTemplate Object\)](#)
- [Type Property \(FieldTemplate Object\)](#)

### Description

- [Changing Data Field Attributes](#)

### Object

- [FieldTemplate Object](#)

## Related

# Hidden Property Example

This example uses the **Hidden** property of the FieldTemplate object to find a data field's format.

```
Dim ABC As Object
Dim MasterItems As Object
Dim Chart As Object
Dim Field_Template As Object
Dim Field_Hidden As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory") ' Add a data field

Field_Hidden = Field_Template.Hidden                   ' Find value of hidden attribute
Select Case Field_Hidden                               ' Report value of hidden attribute
    Case True
        MsgBox "Field is Hidden."
    Case Else
        MsgBox "Field is Visible."
End Select
```

## Name Property (FieldTemplate Object)

Related

Related

Related

### Usage

*FieldTemplateObject.Name = FieldName*

### Description

The **Name** property lets you rename a data field or find the name of a data field. A data field's name appears in the chart next to the field's value. You name a field when you create it with the **Add** method of the FieldTemplates Collection. The **Name** property is read/write.

### Data Type

String

### Value

The name of the data field

### ABC Equivalent

The **Name** property is equivalent to choosing Setup in the ABC Fields menu, selecting a data field, changing the name for the field in the Setup Fields dialog box, and clicking OK.

## Related Topics

### Language Elements

- [Accumulation Property](#)
- [Add Method \(FieldTemplates Collection\)](#)
- [Format Property](#)
- [Hidden Property](#)
- [Name property \(Application object\)](#)
- [Name property \(Chart object\)](#)
- [Name property \(FieldValue object\)](#)
- [Name property \(Font object\)](#)
- [Type Property \(FieldTemplate Object\)](#)

### Description

- [Changing Data Field Attributes](#)

### Object

- [FieldTemplate Object](#)



## Type Property (FieldTemplate Object)

Related

Related

Related

**Usage** *FieldTemplateObject.Type = FieldType*

**Description** The **Type** property of the FieldTemplate object lets you find or set the type of a data field. It is identical with the **Type** property of the FieldValue object. The **Type** property is read/write.

**Data Type** Integer

**Value** The **Type** property uses the values shown in the following table.

Value	Description
0	Text
1	Duration
2	Date
3	Currency
4	Percent
5	Number

**ABC Equivalent** The **Type** property is equivalent to choosing Setup in the Fields menu, clicking the arrow to the right of the Field Type text box, and clicking the type you want for the field.

## Related Topics

### Language Elements

[Type Property \(ABC Object\)](#)

[Type Property \(Chart Object\)](#)

[Type Property \(FieldValue Object\)](#)

[Type Property \(Line Object\)](#)

### Description

[Changing Data Field Attributes](#)

### Object

[FieldTemplate Object](#)

**Related**

## Type Property (FieldTemplate Object) Example

This example uses the **Type** property of the FieldTemplate object to get and display the type of a data field.

```
Dim ABC As Object, MasterItems As Object, Chart As Object
Dim Field_Template As Object
Dim Field_Name As String
Dim Field_Type As Single

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart
Set MasterItems = Chart.MasterItems
Set Field_Template = Chart.FieldTemplates.Add("Inventory") ' Add a data field

Field_Type = Field_Template.Type                       ' Get type

Select Case Field_Type                                 ' Display type
    Case 0
        MsgBox "Field Type = Text"
    Case 1
        MsgBox "Field Type = Duration"
    Case 2
        MsgBox "Field Type = Date"
    Case 3
        MsgBox "Field Type = Currency"
    Case 4
        MsgBox "Field Type = Percent"
    Case 5
        MsgBox "Field Type = Number"
End Select
```

## FieldTemplates Collection

**Description** The FieldTemplates collection is below the Chart object. Below the FieldTemplates collection are the FieldTemplate objects. You can have multiple FieldTemplate objects in the collection.

### Properties

---

[Application](#)

[Count](#)

[Parent](#)

### Methods

---

[Add](#)

[DeleteField](#)

[Item](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Add Method (FieldTemplates Collection)

Related

Related

Related

<b>Usage</b>	<i>FieldTemplatesCollection.Add</i> ( <i>FieldName</i> [, <i>FieldType</i> ]) The <i>FieldName</i> element is the name of the data field you want to create. The <i>FieldType</i> element is optional and defines the type of the data field.														
<b>Description</b>	The <b>Add</b> method of the FieldTemplates collection lets you create a data field. The field created is added to the FieldTemplates collection. You provide the name for the field and, optionally, the type of field to create. The method returns the data field created.														
<b>Data Type</b>	Object														
<b>Value</b>	The <b>Add</b> method returns the newly created FieldTemplate object. The values for the <i>FieldType</i> element (the types of data field) are in the following table. <table><thead><tr><th>Value</th><th>Type</th></tr></thead><tbody><tr><td>0</td><td>Text</td></tr><tr><td>1</td><td>Duration</td></tr><tr><td>2</td><td>Date</td></tr><tr><td>3</td><td>Currency</td></tr><tr><td>4</td><td>Percent</td></tr><tr><td>5</td><td>Number (default if the element is omitted)</td></tr></tbody></table>	Value	Type	0	Text	1	Duration	2	Date	3	Currency	4	Percent	5	Number (default if the element is omitted)
Value	Type														
0	Text														
1	Duration														
2	Date														
3	Currency														
4	Percent														
5	Number (default if the element is omitted)														
<b>ABC Equivalent</b>	The <b>Add</b> method is equivalent to opening the ABC Fields menu, choosing the Setup command, entering a name for the field in the Setup Fields dialog box, selecting a type for the field, and clicking OK.														

## Related Topics

### Language Elements

[AccumulationMethod Property](#)

[Add Method \(Charts Collection\)](#)

[Format Property](#)

[Hidden Property](#)

[Name Property \(FieldTemplate Object\)](#)

[Type Property \(FieldTemplate Object\)](#)

### Description

[Adding Data Fields to a Chart](#)

### Object

[FieldTemplates Collection](#)

**Related**

## Add Method and Count Property (FieldTemplates Collection) Example

This example uses the **Add** method and **Count** property of the FieldTemplates collection to create and count data fields in a chart. The Count properties of other objects and collections work the same way.

```
Dim ABC As Object, Chart As Object
Dim Field_One As Object, Field_Two As Object, Field_Three As Object
Dim Field_Templates_Collection As Object
Dim ABC_Field_Count As Long

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Field_Templates_Collection = Chart.FieldTemplates

Set Field_One = Field_Templates_Collection.Add("Fred", 1)   ' Create fields with
Set Field_Two = Field_Templates_Collection.Add("Wilma", 2)  ' ("Name", FieldType)
Set Field_Three = Field_Templates_Collection.Add("Barney", 3)

ABC_Field_Count = Field_Templates_Collection.Count        ' Get count of fields in chart

MsgBox "There are " + CStr(ABC_Field_Count) + " data fields in the chart."
```

## DeleteField Method

Related

Related

Related

**Usage** *FieldTemplatesCollection.DeleteField FieldTemplateObject*

The *FieldTemplateObject* element is the data field that you want to delete.

**Description** The **DeleteField** method lets you delete a data field. This removes the data field from every shape in the chart. Any values that were in the field are deleted.

**ABC Equivalent** The **DeleteField** method is equivalent to choosing Setup in the ABC Fields menu, selecting a data field in the Field list box, and clicking Delete.

## Related Topics

### Language Elements

[AccumulationMethod Property](#)

[Format Property](#)

[Hidden Property](#)

[Name Property \(FieldTemplate Object\)](#)

[Type Property \(FieldTemplate Object\)](#)

### Description

[Deleting Data Fields from a Chart](#)

### Object

[FieldTemplates Collection](#)

**Related**

## DeleteField, Item Methods Example

This example uses the **DeleteField** method and the **Item** method of the FieldTemplates collection to identify and delete a field.

```
Dim ABC As Object, Chart As Object
Dim Shapel As Object
Dim Field1 As Object, Field2 As Object, Field3 As Object
Dim userInput As String, Msg1 As String

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New                                                 ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shapel = Chart.DrawShape("Operation")              ' Draw a shape
Shapel.Selected = True                                 ' Select the shape

' Add fields to the field template ("Name", FieldType)
Set Field1 = Chart.FieldTemplates.Add("1: Fred", 5)
Set Field2 = Chart.FieldTemplates.Add("2: Wilma", 4)
Set Field3 = Chart.FieldTemplates.Add("3: Barney", 3)
ABC.FieldViewer.Visible = True                         ' Show the field viewer

' Ask user to input the number of the field to be deleted
Msg1 = "Please Enter the number of the field to be deleted (1, 2, or 3)."
UserInput = InputBox$(Msg1; "Delete Field Box")

' Use Item(index) method to delete the correct field
Select Case UserInput
    Case "1"
        Chart.FieldTemplates.DeleteField Chart.FieldTemplates.Item(1)
        MsgBox "Field 1 deleted."
    Case "2"
        Chart.FieldTemplates.DeleteField Chart.FieldTemplates.Item(2)
        MsgBox "Field 2 deleted."
    Case "3"
        Chart.FieldTemplates.DeleteField Chart.FieldTemplates.Item(3)
        MsgBox "Field 3 deleted."
End Select
```

## Item Method (FieldTemplates Collection)

Related

Related

Related

<b>Usage</b>	<i>FieldTemplatesCollection.Item</i> ({ <i>Count</i>   <i>FieldName</i> } [, <i>FieldType</i> ]) The first element is either a <i>Count</i> or a <i>FieldName</i> . The <i>Count</i> element is the number of the item within the collection. The <i>FieldName</i> element is the name of the field. The second element, which is optional, is the type. The <i>FieldType</i> element lets you specify the type of element to be returned.
<b>Description</b>	Use the <b>Item</b> method to access FieldTemplate objects, or data fields, within the FieldTemplates collection.
<b>Data Type</b>	Object
<b>Value</b>	Returns the next valid FieldTemplate object (data field), in the collection. If that object does not exist, the method returns Null.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Accumulation Property](#)

[Item Method \(Charts Collection\)](#)

[Item Method \(FieldValues Collection\)](#)

[Item Method \(Menu Collection\)](#)

[Item Method \(Objects Collection\)](#)

[Value Property](#)

### Description

[Working with Data Field Values](#)

### Object

[FieldTemplates Collection](#)

## FieldValue Object

**Description** The FieldValue object is below the FieldValues collection. You can have multiple FieldValue objects.

### Properties

[Application](#)

[Day](#)

[FieldTemplate](#)

[FormattedValue](#)

[IsEmpty](#)

[Month](#)

[Name](#)

[Parent](#)

[Type](#)

[Value](#)

[Year](#)

### Methods

[Empty](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

[VBX Event Variables](#)

## FieldTemplate Property

Related

Related

Related

**Usage** *FieldValueObject*.FieldTemplate

**Description** You use the **FieldTemplate** property to find the **FieldTemplate** object that corresponds to the field value. The **FieldTemplate** property is read only, but the properties from the object it returns are read/write.

**Data Type** Object

**Value** The FieldTemplate object

**ABC Equivalent** None

## Related Topics

### Description

[Adding Data Fields to a Chart](#)

### Object

[FieldValue Object](#)

**Related**

## FieldTemplate Property Example

This example uses the **FieldTemplate** property of the FieldValue object to make data fields opaque.

```
Sub Command1_Click ()

    Dim ABC As Object, Chart As Object
    Dim Field1 As Object
    Dim Shape1 As Object
    Dim Field_Template As String

    Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
    Set Chart = ABC.ActiveChart                       ' Get the active chart

    Set Field1 = Chart.FieldTemplates.Add("Inventory", 5)  ' Create a field

    Set Shape1 = Chart.DrawShape("Delay")             ' Draw a Delay shape

    Shape1.FieldValues.Item("Inventory").Value = "300,000" ' Enter text in the field

    ' Get the FieldTemplate Object corresponding to this field.
    Field_Template = Shape1.FieldValues.Item("Inventory").FieldTemplate
    MsgBox Field_Template
End Sub
```

## FormattedValue Property

Related

Related

Related

**Usage** *FieldValueObject*.**FormattedValue**

**Description** The **FormattedValue** property lets you find the text string that represents the contents of the field. The **FormattedValue** property is read only.

**Data Type** String

**Value** The text string that represents the contents of the field

**ABC Equivalent** None

## Related Topics

### Description

[Working with Data Field Values](#)

### Object

[FieldValue Object](#)

**Related**

## FormattedValue, FieldTemplates Properties Example

This example uses the **FormattedValue** property of the FieldValue object and the FieldTemplates property of the Chart object to find the formatted value contained in a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object
Dim Field_Formatted_Value As String

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory", 5)   ' Create a field

Set Shape1 = Chart.DrawShape("Delay")                  ' Draw a Delay shape

Shape1.FieldValues.Item("Inventory").Value = "300,000" ' Enter text in the field

Field_Formatted_Value = Shape1.FieldValues.Item("Inventory").FormattedValue
MsgBox Field_Formatted_Value
```

## IsEmpty Property

Related

Related

Related

**Usage** *FieldValueObject.IsEmpty*

**Description** The **IsEmpty** property lets you find whether a data field contains any values. The **IsEmpty** property is read only.

**Data Type** Integer (Boolean)

**Value** True means the data field is empty; False means it contains a value.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Empty Method](#)

[Item Method \(FieldValues Collection\)](#)

### Description

[Working with Data Field Values](#)

### Object

[FieldValue Object](#)

## Name Property (FieldValue Object)

Related

Related

Related

<b>Usage</b>	<i>FieldValueObject.Name = FieldName</i>
<b>Description</b>	The <b>Name</b> property of the FieldValue object lets you find the name of a data field. The field was named when you created it with the <b>Add</b> method of the FieldTemplates Collection. The <b>Name</b> property is read only.
<b>Data Type</b>	String
<b>Value</b>	The name of the data field
<b>ABC Equivalent</b>	The <b>Name</b> property is equivalent to choosing Setup in the ABC Fields menu, selecting a data field, changing the name for the field in the Setup Fields dialog box, and clicking OK.

## Related Topics

### Language Elements

- [AccumulationMethod Property](#)
- [Add Method \(FieldTemplates Collection\)](#)
- [Format Property](#)
- [Hidden Property](#)
- [Name Property \(Application Object\)](#)
- [Name Property \(Chart Object\)](#)
- [Name Property \(FieldTemplate Object\)](#)
- [Name Property \(FieldValue Object\)](#)
- [Name Property \(Font Object\)](#)
- [Type Property \(FieldValue Object\)](#)

### Description

- [Changing Data Field Attributes](#)

### Object

- [FieldValue Object](#)

## Related

### Name Property (FieldValue Object) Example

This example uses the **Name** property of the FieldValue object to find the name of a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object
Dim Field_Name As String
Dim Field_Type

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory")     ' Create a field
Field1.Type = 0                                       ' Make the field's type text

Set Shape1 = Chart.DrawShape("Delay")                 ' Draw a Delay shape

Field_Name = Shape1.FieldValues.Item(1).Name          ' Get the name of the field

MsgBox Field_Name
```

## Type Property (FieldValue Object)

Related

Related

Related

### Usage

*FieldValueObject.Type = FieldType*

### Description

The **Type** property of the FieldValue object lets you find or set the type of a data field. It is identical with the **Type** property of the FieldTemplate object. The **Type** property is read/write.

### Data Type

Integer

### Value

The **Type** property uses the values shown in the following table.

Value	Description
-------	-------------

0	Text
---	------

1	Duration
---	----------

2	Date
---	------

3	Currency
---	----------

4	Percent
---	---------

5	Number
---	--------

### ABC Equivalent

The **Type** method is equivalent to choosing Setup in the Fields menu, clicking the arrow to the right of the Field Type text box, and clicking the type you want for the field.

## Related Topics

### Language Elements

[Type Property \(Chart Object\)](#)

[Type Property \(FieldTemplate Object\)](#)

[Type Property \(Line Object\)](#)

[Type Property \(Object Object\)](#)

### Description

[Changing Data Field Attributes](#)

### Object

[FieldValue Object](#)

## Related

### Type Property (FieldValue Object) Example

This example uses the **Type** property of the FieldValue object to find the type of a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object
Dim Field_Type As Single

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory", 4)   ' Create a field
MsgBox "Field created with format 4."

Field1.Type = 0                                       ' Change the field's type to text

Set Shape1 = Chart.DrawShape("Delay")                  ' Draw a Delay shape

Field_Type = Shape1.FieldValues.Item(1).Type          ' Get type of field just created
MsgBox "The field has changed to format " + CStr(Field_Type) + "."
```

## Value Property

Related

Related

Related

<b>Usage</b>	<i>FieldValueObject.Value = Value</i>
<b>Description</b>	The <b>Value</b> property lets you find or set the value of a data field item of a shape. The <b>Value</b> property is read/write.
<b>Data Type</b>	Variant
<b>Value</b>	The value of the data field item
<b>ABC Equivalent</b>	The <b>Value</b> property is equivalent to selecting a shape, opening the Field Viewer, and entering a value for a data field.

## Related Topics

### Language Elements

[AccumulationMethod Property](#)

[Item Method \(FieldValues Collection\)](#)

### Description

[Working with Data Field Values](#)

### Object

[FieldValue Object](#)

## Empty Method

Related

Related

Related

**Usage** *FieldValueObject.Empty*

**Description** The **Empty** method lets you remove all values from a data field. After you use the method, the **IsEmpty** property of the FieldValue object is True.

**ABC Equivalent** The **Empty** method is equivalent to removing the value from a data field using the field viewer.

## **Related Topics**

### **Language Elements**

[IsEmpty Property](#)

[Value Property](#)

### **Description**

[Working with Data Field Values](#)

### **Object**

[FieldValue Object](#)

## Related

# Empty Method Example

This example uses the **Empty** method of the FieldValue object to remove the value from a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shapel As Object
Dim Field_Formatted_Value As String

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory", 5)   ' Create a field

Set Shapel = Chart.DrawShape("Delay")                  ' Draw a Delay shape

Shapel.FieldValues.Item("Inventory").Value = "300,000" ' Enter text in the field
MsgBox "Formatted field value is " + Shapel.FieldValues("Inventory").FormattedValue + "."

Shapel.FieldValues.Item("Inventory").Empty             ' Empty the field
MsgBox "Empty field value is " + Shapel.FieldValues.Item("Inventory").FormattedValue
```

## Day Property

Related

Related

Related

**Usage** *FieldValueObject.Day = Number*

**Description** You use the **Day** property to find or set the day of the month in a data field. If the data field does not contain a valid date (for example, if it is not a Date field), then the **Day** property is equal to 0. The **Day** property is read/write.

**Data Type** Integer

**Value** A number from 1 to 31

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[Month Property](#)

[Year Property](#)

### **Description**

[Working with Data Field Values](#)

### **Object**

[FieldValue Object](#)

## Related

# Day, Month, Year Properties Example

This example uses the **Day**, **Month**, and **Year** properties of the FieldValue object to find and display the dates in a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shape1 As Object

Dim Field_Day
Dim Field_Month
Dim Field_Year

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Closing Date", 2) ' Create a date field

Set Shape1 = Chart.DrawShape("Delay")                  ' Draw a Delay shape

Shape1.FieldValues.Item("Closing Date").Value = "6/28/95" ' Enter text in the field

Field_Day = Shape1.FieldValues.Item("Closing Date").Day ' Get the date values
Field_Month = Shape1.FieldValues.Item("Closing Date").Month
Field_Year = Shape1.FieldValues.Item("Closing Date").Year

MsgBox "The day listed in the Closing Date is " + Field_Day + " ."
MsgBox "The month listed in the Closing Date is " + Field_Month + " ."
MsgBox "The year listed in the Closing Date is " + Field_Year + " ."
```

## Month Property

Related

Related

Related

**Usage** *FieldValueObject.Month = Number*

**Description** You use the **Month** property to find or set the month in a data field. If the data field does not contain a valid date (for example, if it is not a Date field), then the **Month** property is equal to 0. The **Month** property is read/write.

**Data Type** Integer

**Value** A number from 1 to 31

**ABC Equivalent** None

## Related Topics

### Language Elements

[Day Property](#)

[Year Property](#)

### Description

[Working with Data Field Values](#)

### Object

[FieldValue Object](#)

## Year Property

Related

Related

Related

**Usage** *FieldValueObject.Year = Number*

**Description** You use the **Year** property to find or set the month in a data field. If the data field does not contain a valid date (for example, if it is not a Date field), then the **Year** property is equal to 0. The **Year** property is read/write.

**Data Type** Integer

**Value** A number 1900 or larger

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[Day Property](#)

[Month Property](#)

### **Description**

[Working with Data Field Values](#)

### **Object**

[FieldValue Object](#)

## FieldValues Collection

**Description** The FieldValues collection is below the Object object. Below the FieldValues collection are the FieldValue objects. You can have multiple FieldValue objects in the FieldValues collection.

### Properties

[Application](#)

[Count](#)

[Parent](#)

### Methods

[Item](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Item Method (FieldValues Collection)

Related

Related

Related

### Usage

*FieldValuesCollection.Item* ({*Count* | *FieldName*} [, *FieldType*])

The first element is either a *Count* or a *FieldName*.

The *Count* element is the number of the item within the collection.

The *FieldName* element is the name of the field.

The *FieldType* element, which is optional, lets you specify the type of element to be returned.

### Description

Use the **Item** method of the FieldValues collection to access FieldValue objects, or data fields, within the FieldValues collection.

### Data Type

Object

### Value

The next valid FieldValue object, or data field, in the collection. If that object does not exist, the method returns Null.

### ABC Equivalent

None

## Related Topics

### Language Elements

[IsEmpty Property](#)

[Item method \(Charts collection\)](#)

[Item method \(FieldTemplates collection\)](#)

[Item method \(Menu collection\)](#)

[Item method \(Objects collection\)](#)

[Value Property](#)

### Description

[Working with Data Field Values](#)

### Object

[FieldValues collection](#)

## Related

# Item Method (FieldValues Collection), Value Property, and IsEmpty Property Example

This example uses the **Item** method of the FieldValues collection and the **Value** property and **IsEmpty** property of the FieldValue object to find whether a data field contains a value.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object
Dim Shape1 As Object
Dim Field_Empty As Integer

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Inventory")     ' Create a field

Field1.Type = 0                                       ' Make the field's type text

Set Shape1 = Chart.DrawShape("Delay")                  ' Draw a Delay shape

Shape1.FieldValues.Item("Inventory").Value = "300,000" ' Enter text in the field

Field_Empty = Shape1.FieldValues.Item("Inventory").IsEmpty
Select Case Field_Empty
    Case True
        MsgBox "Field is empty."
    Case Else
        MsgBox "Field is not empty."
End Select
```

## Font Object

**Description**      The Font object is below the Object object. You can have only one Font object for each Object object.

### **Properties**

---

[Application](#)

[Bold](#)

[Color](#)

[Italic](#)

[Name](#)

[Opaque](#)

[Parent](#)

[Size](#)

[Strikethrough](#)

[Underline](#)

### **Methods**

---

There are no methods for the Font object.

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Bold Property

Related

Related

Related

### Usage

*FontObject*.**Bold** = {True | False}

### Description

You use the **Bold** property to change text to be bold, including text in shape numbers using the **NumberFont** property of the Chart object and note text using the **NoteFont** property of the Shape object. The **Bold** property is read/write.

### Data Type

Integer (Boolean)

### Value

True means the text is bold; False means the text is not bold.

### ABC Equivalent

The **Bold** property is equivalent to selecting the text you want to affect, clicking the Text tool, and clicking the Bold button.

## **Related Topics**

### **Language Elements**

[NoteFont Property](#)

[NumberFont Property](#)

### **Description**

[Bold, Italic, Underline, and Strikethrough](#)

[Formatting Shape Numbers](#)

[Formatting Note Text](#)

### **Object**

[Font Object](#)

## Color Property (Font Object)

Related

Related

Related

### Usage

*FontObject.Color = Color*

### Description

You use the **Color** property of the Font object to find or set the color of selected text or the text inside a shape object. The **Color** property affects only the foreground color of the text, not the background color. The **Color** property is read/write.

### Data Type

Long

### Value

The color for the text

### ABC Equivalent

The **Color** property of the Font object is equivalent to selecting text or highlighting a portion of text, clicking the Text tool, and clicking the color you want in the Text ribbon.

## **Related Topics**

### **Language Elements**

[BasicColor Method](#)

[Bold Property](#)

[Color Property \(Line Object\)](#)

[Color Property \(Object Object\)](#)

[Italic Property](#)

[MakeRGB Method](#)

[Opaque Property](#)

[Size Property](#)

[TextAlignment Property](#)

[Underline Property](#)

### **Description**

[Color Representation](#)

[Setting Text Colors](#)

[Text Color](#)

### **Object**

[Font Object](#)

## Italic Property

Related

Related

Related

<b>Usage</b>	<i>FontObject.Italic</i> = {True   False}
<b>Description</b>	You use the <b>Italic</b> property to change text to be italic, including text in shape numbers using the <b>NumberFont</b> property of the Chart object and note text using the <b>NoteFont</b> property of the Shape object. The <b>Italic</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the text is <i>italic</i> ; False means the text is not italic.
<b>ABC Equivalent</b>	The <b>Italic</b> property is equivalent to selecting the text you want to affect, clicking the Text tool, and clicking the Italic button.

## Related Topics

### Language Elements

[NoteFont Property](#)

[NumberFont Property](#)

### Description

[Bold, Italic, Underline, and Strikethrough](#)

[Formatting Shape Numbers](#)

[Formatting Note Text](#)

### Object

[Font Object](#)

## Name Property (Font Object)

Related

Related

Related

<b>Usage</b>	<i>FontObject.Name = FontName</i>
<b>Description</b>	You use the <b>Name</b> property of the Font object to set the typeface name for the font, such as "Arial" or "Roman." The <b>Name</b> property is read/write.
<b>Data Type</b>	String
<b>Value</b>	The typeface name
<b>ABC Equivalent</b>	The <b>Name</b> property is equivalent to selecting the text you want to affect, clicking the Text tool, clicking down arrow to the right of the Font box, and clicking the font you want.

## Related Topics

### Language Elements

[Name Property \(Application Object\)](#)

[Name Property \(Chart Object\)](#)

[Name Property \(FieldTemplate Object\)](#)

[Name Property \(FieldValue Object\)](#)

### Description

[Formatting Text](#)

[Formatting Shape Numbers](#)

[Formatting Note Text](#)

[Text Typeface and Size](#)

### Object

[Font Object](#)



## Opaque Property

Related

Related

Related

<b>Usage</b>	<i>FontObject</i> . <b>Opaque</b> = {True   False}
<b>Description</b>	The <b>Opaque</b> property lets you set or find whether text background is opaque. The <b>Opaque</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the text background is opaque; False means it is transparent.
<b>ABC Equivalent</b>	The <b>Opaque</b> property is equivalent to selecting the text you want to affect, clicking the Text tool, and clicking the Opaque button.

## **Related Topics**

### **Language Elements**

[NoteFont Property](#)

[NumberFont Property](#)

### **Description**

[Formatting Text](#)

[Formatting Shape Numbers](#)

[Formatting Note Text](#)

[Text Background](#)

### **Object**

[Font Object](#)

## Related

# Opaque Property and AttachText Method Example

This example uses the **Opaque** property of the Font object and the **AttachText** method of the Line\_ object to make text opaque and attach text to a line.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object, Text1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Shape1 = Chart.DrawShape("Decision")
Set Shape2 = Chart.DrawShape("Operation")
Set Line1 = Chart.DrawLine(Shape1, Shape2)

Set Text1 = Chart.DrawTextBlock("This way!")
Text1.Font.Opaque = True
Line1.Line_1.AttachText Text1
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' Draw a Decision shape
' Draw an Operation shape
' Draw a line connecting the shapes

' Create a freeform text object
' Make the text's background opaque
' Attach the text object to the line
```

## Size Property

Related

Related

Related

**Usage** *FontObject.Size = FontSize*

**Description** You use the **Size** property of the Font object to set the typeface size in points. The **Size** property is read/write.

**Data Type** Long

**Value** The point size

**ABC Equivalent** The **Size** property is equivalent to selecting the text you want to affect, clicking the Text tool, and choosing the size you want.

## **Related Topics**

### **Description**

[Formatting Text](#)

[Formatting Shape Numbers](#)

[Formatting Note Text](#)

[Text Typeface and Size](#)

### **Object**

[Font Object](#)

## Strikethrough Property

Related

Related

Related

<b>Usage</b>	<i>FontObject.Strikethrough</i> = {True   False}
<b>Description</b>	You use the <b>Strikethrough</b> property to change text to have a line through it, including text in shape numbers using the <b>NumberFont</b> property of the Chart object and note text using the <b>NoteFont</b> property of the Shape object. The <b>Strikethrough</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the text is <del>strikethrough</del> ; False means the text is not <del>strikethrough</del> .
<b>ABC Equivalent</b>	The <b>Strikethrough</b> property is equivalent to selecting the text you want to affect, clicking the Text tool, and clicking the Strikethrough button.

## Related Topics

### Language Elements

[NoteFont Property](#)

[NumberFont Property](#)

### Description

[Bold, Italic, Underline, and Strikethrough](#)

[Formatting Shape Numbers](#)

[Formatting Note Text](#)

### Object

[Font Object](#)

## Underline Property

Related

Related

Related

### Usage

*FontObject.Underline* = {True | False}

### Description

You use the **Underline** property to underline selected text, including text in shape numbers using the **NumberFont** property of the Chart object and note text using the **NoteFont** property of the Shape object. The **Underline** property is read/write.

### Data Type

Integer (Boolean)

### Value

True means the text is underlined; False means the text is not underlined.

### ABC Equivalent

The **Underline** property is equivalent to selecting the text you want to affect, clicking the Text tool, and clicking the Underline button.

## Related Topics

### Language Elements

[NoteFont Property](#)

[NumberFont Property](#)

### Description

[Bold, Italic, Underline, and Strikethrough](#)

[Formatting Shape Numbers](#)

[Formatting Note Text](#)

[Bold, Italic, Underline, and Strikethrough](#)

### Object

[Font Object](#)

## Line\_ Object

**Description** The Line\_ object is below the Object object. You can have only one Line\_ object for each Object object. If the Object object is a shape, this object is a meaningless placeholder.

### Properties

---

[Application](#)  
[Color](#)  
[DestArrowColor](#)  
[DestArrowSize](#)  
[DestArrowStyle](#)  
[Destination](#)  
[DestinationDirection](#)  
[Parent](#)  
[Source](#)  
[SourceArrowColor](#)  
[SourceArrowSize](#)  
[SourceArrowStyle](#)  
[SourceDirection](#)  
[StemColor](#)  
[StemStyle](#)  
[StemWidth](#)  
[Type](#)

### Methods

---

[AttachText](#)  
[ReconnectDest](#)  
[ReconnectSource](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Color Property (Line\_ Object)

Related

Related

Related

### Usage

*Line\_Object.Color = Color*

### Description

You use the **Color** property of the Line\_ object to set the color for lines, including the line ends and the stem, or find the stem color of lines. The **Color** property colors the entire line, including the ends. The **Color** property is read/write.

### Data Type

Long

### Value

The color for a line object

### ABC Equivalent

The **Color** property of the Line\_ object is equivalent to selecting a line, clicking the Line tool in the toolbox, pressing Shift, clicking the Apply buttons next to the Line Start, Line Style, and Line End buttons in the ribbon, and then clicking the color you want.

## Related Topics

### Language Elements

[BasicColor Method](#)

[Color Property \(Font Object\)](#)

[Color Property \(Object Object\)](#)

[DestArrowColor Property](#)

[MakeRGB Method](#)

[SourceArrowColor Property](#)

[StemColor Property](#)

### Description

[Color Representation](#)

[Setting Line Colors](#)

[Line Color](#)

### Object

[Line\\_ Object](#)

## Related

# Color, DestArrowStyle, SourceArrowStyle Properties (Line\_ Object) Example

This example uses the **Color** property, the **DestArrowStyle** property, and the **SourceArrowStyle** property of the **Line\_** object to set the color, destination arrow style, and source arrow style for a line.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart

Set Shape1 = Chart.DrawShape("Decision")               ' Draw a Decision shape
Set Shape2 = Chart.DrawShape("Operation")              ' Draw an Operation shape
Set Line1 = Chart.DrawLine(Shape1, Shape2)              ' Draw a line connecting the shapes

Line1.Color = ABC.MakeRGB(255, 0, 0)                   ' Make the line red
Line1.Line_.DestArrowStyle = 12                        ' Apply a double arrowhead
Line1.Line_.SourceArrowStyle = 4                       ' Apply a source arrow style
```

## DestArrowColor Property

Related

Related

Related

<b>Usage</b>	<i>Line_Object</i> .DestArrowColor = Color
<b>Description</b>	You use the <b>DestArrowColor</b> property to find or set the color of the destination arrow of a line. The <b>DestArrowColor</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The color of the destination arrow of a line
<b>ABC Equivalent</b>	The <b>DestArrowColor</b> property is equivalent to selecting a shape, clicking the Line tool in the toolbox, clicking the Apply button next to the End Style button in the ribbon, and then clicking the color you want.

## Related Topics

### Language Elements

[BasicColor Method](#)

[Color Property \(Line\\_ Object\)](#)

[DestArrowSize Property](#)

[DestArrowStyle Property](#)

[MakeRGB Method](#)

[SourceArrowColor Property](#)

[StemColor Property](#)

### Description

[Color Representation](#)

[Setting Line Colors](#)

[Line Color](#)

### Object

[Line\\_ Object](#)

## DestArrowSize Property

Related

Related

Related

<b>Usage</b>	<i>Line_Object</i> .DestArrowSize = Size
<b>Description</b>	You use the <b>DestArrowSize</b> property to find or set the destination arrow size. The <b>DestArrowSize</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	Arrow size can vary from 1 (smallest) to 5 (largest).
<b>ABC Equivalent</b>	The <b>DestArrowSize</b> property is equivalent to selecting a line, clicking the Line tool and clicking the apply button in the ribbon next to the End Style button in the ribbon, and then choosing a number in the Width box.

## Related Topics

### Language Elements

[DestArrowColor Property](#)

[DestArrowStyle Property](#)

[SourceArrowSize Property](#)

[StemWidth Property](#)

### Description

[Line Width](#)

### Object

[Line Object](#)

## DestArrowStyle Property

Related

Related

Related

Related

### Usage

*Line\_Object*.DestArrowStyle = StyleNumber

### Description

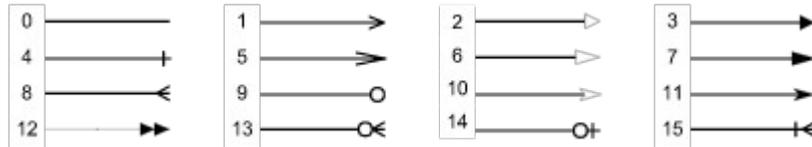
You use the **DestArrowStyle** property to find or set styles or patterns for line ends, including arrows, circles, and lines. The **DestArrowStyle** property is read/write.

### Data Type

Integer

### Value

You set the **DestArrowStyle** property to 0 for no arrow. The following illustration shows the values of the **DestArrowStyle** property for each available style.



### ABC Equivalent

The **DestArrowStyle** property is equivalent to selecting a line, clicking the Line tool, clicking the End Style button in the ribbon, and then clicking the end style you want.

## Related Topics

### Language Elements

[DestArrowColor Property](#)

[DestArrowStyle Property](#)

[SourceArrowStyle Property](#)

[StemStyle Property](#)

### Description

[End Styles](#)

### Object

[Line Object](#)

## Destination Property

Related

Related

Related

### Usage

*Line\_Object.Destination = Shape*

### Description

You use the **Destination** property to connect lines to shapes. The shapes that lines connect to are stored in the **Source** property and **Destination** property. When lines are unconnected, those properties are empty. The **Destination** property is read/write.

### Data Type

Object

### Value

A shape object that a line is connected to

### ABC Equivalent

The **Destination** property is equivalent to dragging the end of a line into a shape.

## Related Topics

### Language Elements

[ReconnectDest Method](#)

[Source Property](#)

### Description

[Connecting Existing Lines to Shapes](#)

### Object

[Line\\_Object](#)

## Source Property

Related

Related

Related

**Usage** *Line\_Object.Source = Shape*

**Description** You use the **Source** property to connect lines to shapes. The shapes that lines connect to are stored in the **Source** property and **Destination** property. When lines are unconnected, those properties are empty. The **Source** property is read/write.

**Data Type** Object

**Value** A shape object that a line is connected to

**ABC Equivalent** The **Source** property is equivalent to dragging the end of a line into a shape.

## Related Topics

### Language Elements

[ReconnectDest Method](#)

[Destination Property](#)

### Description

[Connecting Existing Lines to Shapes](#)

### Object

[Line\\_Object](#)

## SourceArrowColor Property

Related

Related

Related

<b>Usage</b>	<i>Line_Object.SourceArrowColor = Color</i>
<b>Description</b>	You use the <b>SourceArrowColor</b> property to find or set the color of the source arrow of a line. The <b>SourceArrowColor</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The color of the source arrow of a line
<b>ABC Equivalent</b>	The <b>SourceArrowColor</b> property is equivalent to selecting a shape, clicking the Line tool in the toolbox, clicking the Apply button next to the Start Style button in the ribbon, and then clicking the color you want.

## Related Topics

### Language Elements

[BasicColor Method](#)

[Color Property \(Line\\_ Object\)](#)

[DestArrowColor Property](#)

[MakeRGB Method](#)

[SourceArrowSize Property](#)

[SourceArrowStyle Property](#)

[StemColor Property](#)

### Description

[Color Representation](#)

[Setting Line Colors](#)

[Line Color](#)

### Object

[Line\\_ Object](#)

## SourceArrowSize Property

Related

Related

Related

<b>Usage</b>	<i>Line_Object.SourceArrowSize = Size</i>
<b>Description</b>	You use the <b>SourceArrowSize</b> property to find or set the line width. The <b>SourceArrowSize</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	Arrow size can vary from 1 (smallest) to 5 (largest).
<b>ABC Equivalent</b>	The <b>SourceArrowSize</b> property is equivalent to selecting a line, clicking the Line tool and clicking the apply button in the ribbon next to the Start Style button in the ribbon, and then choosing a number in the Width box.

## Related Topics

### Language Elements

[DestArrowSize Property](#)

[SourceArrowColor Property](#)

[SourceArrowStyle Property](#)

[StemWidth Property](#)

### Description

[Line Width](#)

### Object

[Line Object](#)

## SourceArrowStyle Property

Related

Related

Related

Related

### Usage

*Line\_Object.SourceArrowStyle = StyleNumber*

### Description

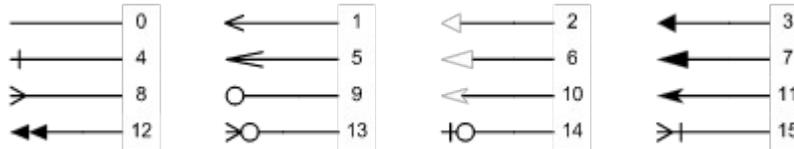
You use the **SourceArrowStyle** property to find or set styles or patterns for line ends, including arrows, circles, and lines. The **SourceArrowStyle** property is read/write.

### Data Type

Integer

### Value

You set the **SourceArrowStyle** property to 0 for no arrow. The following illustration shows the values of the **SourceArrowStyle** property for each available style.



### ABC Equivalent

The **SourceArrowStyle** property is equivalent to selecting a line, clicking the Line tool, clicking the Start Style button in the ribbon, and then clicking the end style you want.

## Related Topics

### Language Elements

[DestArrowStyle Property](#)

[SourceArrowColor Property](#)

[SourceArrowSize Property](#)

[StemStyle Property](#)

### Description

[End Styles](#)

### Object

[Line Object](#)

## StemColor Property

Related

Related

Related

### Usage

*Line\_Object.StemColor = Color*

### Description

The **StemColor** property lets you find or set the color for the stem of a line (see the `MakeRGB` method). The stem is the part of the line between the source and destination arrows. The **StemColor** property is read/write.

### Data Type

Long

### Value

The color for the stem of a line

### ABC Equivalent

The **StemColor** property is equivalent to selecting a line, clicking the Line tool in the toolbox, clicking the Apply button next to the Line Style button in the ribbon, and then clicking the color you want.

## Related Topics

### Language Elements

[BasicColor Method](#)

[Color Property \(Line Object\)](#)

[DestArrowColor Property](#)

[MakeRGB Method](#)

[SourceArrowColor Property](#)

[StemStyle Property](#)

[StemWidth Property](#)

### Description

[Setting Line Colors](#)

[Line Color](#)

### Object

[Line\\_Object](#)

**Related**

## Line\_ Object Properties Example

This example uses the properties of the Line\_ object to set the color, size, and style of the destination arrow, source arrow, and stem of a line.

```
Dim ABC As Object, Chart As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Set Line1 = Chart.DrawFreeLine(4, 4)

Line1.Line_.StemColor = ABC.MakeRGB(0, 0, 255)
Line1.Line_.StemStyle = 3
Line1.Line_.SourceArrowStyle = 13
Line1.Line_.SourceArrowColor = ABC.MakeRGB(255, 0, 0)
Line1.Line_.SourceArrowSize = 3
Line1.Line_.DestArrowColor = ABC.MakeRGB(0, 255, 0)
Line1.Line_.DestArrowSize = 5
Line1.Line_.DestArrowStyle = 5
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' The line's beginning X position
' The line's beginning Y position
' Draw an unconnected line to X=4,Y=4

' Make line's stem blue
' Set stem style
' Set source arrow style
' Make source arrow red
' Make source arrow medium in size
' Make destination arrow green
' Make destination arrow large
' Set destination arrow style
```

## StemStyle Property

Related

Related

Related

### Usage

*Line\_Object.StemStyle = StyleNumber*

### Description

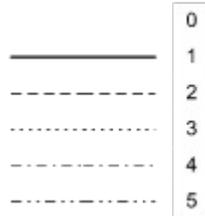
You use the **StemStyle** property to find or set styles or patterns for line stems. The **StemStyle** property is read/write.

### Data Type

Integer

### Value

Set the **StemStyle** property to 0 for an invisible line and 1 for a solid line. The following illustration shows the values of the **StemStyle** property for each available style.



### ABC Equivalent

The **StemStyle** property is equivalent to selecting a line, clicking the Line tool, clicking the Line Style button in the ribbon, and then clicking the line style you want.

## Related Topics

### Language Elements

[DestArrowStyle Property](#)

[SourceArrowStyle Property](#)

[StemColor Property](#)

[StemWidth Property](#)

### Description

[Line Style](#)

### Object

[Line Object](#)

## StemWidth Property

Related

Related

Related

### Usage

*Line\_Object.StemWidth = Width*

### Description

You use the **StemWidth** property to find or set the width of the line stem, excluding the ends. The **StemWidth** property is read/write.

### Data Type

Integer

### Value

Line width can vary from 1 (hairline) to 5 (thickest).

### ABC Equivalent

The **StemWidth** property is equivalent to selecting a line, clicking the Line tool, clicking the apply button in the ribbon next to the Line Style button in the ribbon, and then choosing a number in the Width box.

## Related Topics

### Language Elements

[DestArrowStyle Property](#)

[SourceArrowStyle Property](#)

[StemColor Property](#)

[StemStyle Property](#)

### Description

[Line Width](#)

### Object

[Line Object](#)

**Related**

## StemWidth, Type Properties Example

This example uses the **StemWidth** property and **Type** property of the Line\_ object to make a line wide and determine the type of line drawn.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object
Dim RandomLine As Integer

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Shape1 = Chart.DrawShape("Decision")
Set Shape2 = Chart.DrawShape("Operation")

RandomNum = Int(10 * Rnd)
If RandomNum > 4 Then RandomNum = RandomNum - 5
Chart.CurrentLineRouting = RandomNum

Set Line1 = Chart.DrawLine(Shape1, Shape2, 0, 1)
Line1.Line_.StemWidth = 5

Select Case Line1.Line_.Type
    Case 0
        MsgBox "This is a direct line."
    Case 1
        MsgBox "This is a right angle line."
    Case 2
        MsgBox "This is a curved line."
    Case 3
        MsgBox "This is an org-chart line."
    Case 4
        MsgBox "This is a cause/effect line."
End Select
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' Draw a Decision shape
' Draw an Operation shape

' Generate a random integer
' Eliminate numbers > 4
' Randomly set the line type

' Draw a line connecting the shapes
' Make the line's stem very wide

' Display the type of line used
```

## Type Property (Line Object)

Related

Related

Related

<b>Usage</b>	<i>Line_Object.Type</i>
<b>Description</b>	You use the <b>Type</b> property of the Line_ object to find or set which line routing was used to draw a line. The <b>Type</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The following table describes the values for the Type property.
	<b>Value    Type of Line</b>
	0        Direct
	1        Right angle
	2        Curved
	3        Organization chart
	4        Cause-and-effect
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[CurrentLineRouting Property](#)

[Type Property \(Chart Object\)](#)

[Type Property \(FieldTemplate Object\)](#)

[Type Property \(FieldValue Object\)](#)

[Type Property \(Object Object\)](#)

### Description

[Setting Line Routing](#)

### Object

[Line Object](#)

## AttachText Method

Related

Related

Related

### Usage

*Line\_Object*.**AttachText** *TextObject* [, *LineSegment*]

The *TextObject* element is a text block object that was created using the **DrawTextBlock** method. It is the text to attach to the line.

The *LineSegment* element optionally specifies the segment of the line to which to attach the text.

### Description

You use the **AttachText** method to attach text to a line. You specify the text object to attach and optionally indicate the segment to which the text should be attached.

### Data Type

Integer (Boolean)

### Value

The following table describes each possible value for *LineSegment*.

<b><i>LineSegment</i></b>	<b>Description</b>
---------------------------	--------------------

-2	End
----	-----

-3	Start (default)
----	-----------------

-1	First
----	-------

0	Last
---	------

1 through <i>n</i>	The sequential value of the line segment, where <i>n</i> is the number of segments in the line. For example, 1 is the first segment and 2 is the second segment.
--------------------	--

**ABC Equivalent** The **AttachText** method is equivalent to selecting a text block, dragging it to a line, and snapping it to that line.

## Related Topics

### Language Elements

[DrawTextBlock Method](#)

### Description

[Creating Text Blocks](#)

[Attaching Text to Lines](#)

[Attaching Text to a Line](#)

[Unattaching Text from a Line](#)

### Object

[Line Object](#)

## ReconnectDest Method

Related

Related

Related

<b>Usage</b>	<i>Line_Object.ReconnectDest ShapeObject [, EnterDirection]</i> The <i>ShapeObject</i> element is the shape that the line is to connect to. The <i>EnterDirection</i> element, which is optional, specifies the side where the line should enter the shape.										
<b>Description</b>	You use the <b>ReconnectDest</b> method to connect an existing line to a shape or to change the side where a line enters a shape. You specify the shape that the line enters and, optionally, the side of the shape where the line enters.										
<b>Data Type</b>	Integer (Boolean)										
<b>Value</b>	True means the reconnection was successful; False means it was not successful. The following table shows the values of the <i>EnterDirection</i> element and their meanings. <table><thead><tr><th>Value</th><th>Direction</th></tr></thead><tbody><tr><td>0</td><td>North</td></tr><tr><td>1</td><td>East</td></tr><tr><td>2</td><td>South</td></tr><tr><td>3</td><td>West</td></tr></tbody></table>	Value	Direction	0	North	1	East	2	South	3	West
Value	Direction										
0	North										
1	East										
2	South										
3	West										
<b>ABC Equivalent</b>	The <b>ReconnectDest</b> method is equivalent to dragging a line end into a shape.										

## Related Topics

### Language Elements

[ReconnectSource Method](#)

### Description

[Connecting Existing Lines to Shapes](#)

### Object

[Line\\_Object](#)

## Related

# ReconnectDest, ReconnectSource Methods Example

This example uses the **ReconnectDest** method and **ReconnectSource** method of the `Line_` object to connect the beginning and end of a line to objects.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Set Line1 = Chart.DrawFreeLine(4, 4)

Set Shape1 = Chart.DrawShape("Terminal")
Set Shape2 = Chart.DrawShape("Connector")
Line1.Line_.ReconnectDest Shape2, 1
Line1.Line_.ReconnectSource Shape1, 0
Connector
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' The line's beginning X position
' The line's beginning Y position
' Draw an unconnected line to X=4,Y=4

' Draw a Terminal shape as destination
' Draw a Connector shape as the source
' Connect end of line to bottom of Terminal
' Connect beginning of line to top of
```

## ReconnectSource Method

Related

Related

Related

<b>Usage</b>	<i>Line_Object.ReconnectSource ShapeObject [, ExitDirection]</i> The <i>ShapeObject</i> element is the shape that the line is to connect to. The <i>ExitDirection</i> element, which is optional, specifies the side where the line leaves the shape.										
<b>Description</b>	You use the <b>ReconnectSource</b> method to connect an existing line to a shape or to change the side where a line leaves a shape. You specify the shape that the line leaves and, optionally, the side of the shape where the line leaves.										
<b>Data Type</b>	Integer (Boolean)										
<b>Value</b>	True means the reconnection was successful; False means it was not successful. The following table shows the values of the <i>ExitDirection</i> element and their meanings. <table><thead><tr><th>Value</th><th>Direction</th></tr></thead><tbody><tr><td>0</td><td>North</td></tr><tr><td>1</td><td>East</td></tr><tr><td>2</td><td>South</td></tr><tr><td>3</td><td>West</td></tr></tbody></table>	Value	Direction	0	North	1	East	2	South	3	West
Value	Direction										
0	North										
1	East										
2	South										
3	West										
<b>ABC Equivalent</b>	The <b>ReconnectSource</b> method is equivalent to dragging a line end into a shape.										

## Related Topics

### Language Elements

[ReconnectDest Method](#)

### Description

[Connecting Existing Lines to Shapes](#)

### Object

[Line\\_Object](#)

## DestinationDirection Property

Related

Related

Related

### Usage

*Line\_Object.DestinationDirection = EnterDirection*

### Description

You use the **DestinationDirection** property of the *Line\_* object to set or find the side at which a line drawn between two shapes will enter the ending shape. The line enters at the center of the side. The **DestinationDirection** property is read/write.

### Data Type

Integer

### Value

The side of the ending shape into which a connecting line will enter. The following table shows the values of the *EnterDirection* element and their meanings.

Value	Direction
-------	-----------

0	North
1	East
2	South
3	West

**ABC Equivalent** None

## Related Topics

### Language Elements

[SourceDirection Property](#)

### Description

[Drawing Lines that Connect Shapes](#)

### Object

[Line Object](#)

## Related

### DestinationDirection, SourceDirection Properties Example

This example uses the **DestinationDirection** method and **SourceDirection** method of the Line\_ object to display the destination and source directions of a line connecting two shapes.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim NewLine As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.New

Chart.MasterItems.HideAll

Chart.DrawPositionX = 1
Chart.DrawPositionY = 2
Set Shape1 = Chart.DrawShape
Chart.DrawPositionX = 4
Set Shape2 = Chart.DrawShape
Set NewLine = Chart.DrawLine(Shape1, Shape2)

MsgBox "Source Direction: " + NewLine.Line_.SourceDirection
MsgBox "Destination Direction: " + NewLine.Line_.DestinationDirection
```

```
' Start ABC
' Make ABC visible
' Make a new chart

' Get Master Items out of the way

' Draw 2 shapes and connect...
' ...them with a line
```

## SourceDirection Property Related

Related

Related

**Usage** *Line\_Object.SourceDirection = ExitDirection*

**Description** You use the **SourceDirection** property of the Line\_ object to set or find the side at which a line drawn between two shapes will leave the starting shape. The line leaves at the center of the side. The **SourceDirection** property is read/write.

**Data Type** Integer

**Value** The side of the starting shape from which a connecting line will exit. The following table shows the values of the *ExitDirection* element and their meanings.

Value	Direction
0	North
1	East
2	South
3	West

**ABC Equivalent** None

## Related Topics

### Language Elements

[DestinationDirection Property](#)

### Description

[Drawing Lines that Connect Shapes](#)

### Object

[Line Object](#)

## MasterItems Object

**Description** The MasterItems object is below the Chart object. You can have only one MasterItems object.

### Properties

---

[Application](#)  
[ChartName](#)  
[ChartNameShown](#)  
[Date](#)  
[DateShown](#)  
[DateStyle](#)  
[HideAll](#)  
[Logo](#)  
[LogoPathname](#)  
[LogoShown](#)  
[PageNumber](#)  
[PageNumberShown](#)  
[Parent](#)  
[Range](#)  
[Text1](#)  
[Text1Shown](#)  
[Text2](#)  
[Text2Shown](#)  
[Time](#)  
[TimeShown](#)

### Methods

---

[ShowAll](#)  
[UpdateDateAndTime](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## ChartName Property

Related

Related

Related

**Usage** *MasterItemsObject.ChartName*

**Description** You use the **ChartName** property to find the Chart Name master item object for the chart. The **ChartName** property is read only, but the properties from the object it returns are read/write.

**Data Type** Object

**Value** The chart object

**ABC Equivalent** None

## Related Topics

### Language Elements

[ChartNameShown Property](#)

[HideAll Method](#)

[ShowAll Method](#)

[UpdateDateAndTime Method](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Date Property

Related

Related

Related

**Usage** *MasterItemsObject.Date*

**Description** You use the **Date** property to find or set the Date master item properties. The **Date** property is read only, but the properties from the object it returns are read/write.

**Data Type** Object

**Value** The Date object

**ABC Equivalent** None

## Related Topics

### Language Elements

[DateShown Property](#)

[DateStyle Property](#)

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[UpdateDateAndTime Method](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Logo Property

Related

Related

Related

**Usage** *MasterItemsObject*.**Logo**

**Description** You use the **Logo** property to find the Logo master item. You use the **LogoPathname** property to make the logo appear. The **Logo** property is read only, but the properties from the object it returns are read/write.

**Data Type** Object

**Value** The logo object

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[HideAll Method](#)

[LogoPathname Property](#)

[LogoShown Property](#)

[Range Property](#)

[ShowAll Method](#)

### **Description**

[Displaying Master Items](#)

### **Object**

[MasterItems Object](#)

## PageNumber Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject</i> . <b>PageNumber</b>
<b>Description</b>	The <b>PageNumber</b> property lets you find the page number included in the MasterItems object. The <b>PageNumber</b> property is read only, but all the properties from the object it returns are read/write.
<b>Data Type</b>	Object
<b>Value</b>	The page number included in the MasterItems object
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[MasterItems Property](#)

[PageNumberShown Property](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Text1 Property

Related

Related

Related

**Usage** *MasterItemsObject.Text1*

**Description** You use the **Text1** property to find the Text1 master item. The **Text1** property is read only, but the properties from the object it returns are read/write.

**Data Type** Object

**Value** The Text1 object

**ABC Equivalent** None

## Related Topics

### Language Elements

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[Text1Shown Property](#)

[Text2 Property](#)

[Text2Shown Property](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Related

### Text1, Text2 Properties Example

This example uses the **Text1** method and **Text2** method of the MasterItems object to put text into the text 1 and text 2 master items.

```
Dim ABC As Object, Chart As Object, MasterItems As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

MasterItems.Text1.Text = "This is the Text1 field"     ' Put text in text fields
MasterItems.Text2.Text = "This is the Text2 field"
```

## Text2 Property

Related

Related

Related

**Usage** *MasterItemsObject.Text2*

**Description** You use the **Text2** property to find the Text2 master item. The **Text2** property is read only, but the properties from the object it returns are read/write.

**Data Type** Object

**Value** The Text2 object

**ABC Equivalent** None

## Related Topics

### Language Elements

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[Text1 Property](#)

[Text1Shown Property](#)

[Text2Shown Property](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Time Property

Related

Related

Related

**Usage** *MasterItemsObject.Time*

**Description** You use the **Time** property to find the Time master item. The **Time** property is read only, but the properties from the object it returns are read/write.

**Data Type** Object

**Value** The time object

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[TimeShown Property](#)

[UpdateDateAndTime Method](#)

### **Description**

[Displaying Master Items](#)

### **Object**

[MasterItems Object](#)

## DateStyle Property

Related

Related

Related

- Usage** *MasterItemsObject.DateStyle = Value*
- Description** You use the **DateStyle** property to find or set the style of the Date master item. The **DateStyle** property is read/write.
- Data Type** Integer
- Value** The values for the **DateStyle** property are in the following table.
- | <b>Value</b> | <b>Style</b>                |
|--------------|-----------------------------|
| 0            | MM/DD/YY                    |
| 1            | Short text (Jan. 1, 1995)   |
| 2            | Long text (January 1, 1995) |
- ABC Equivalent** The **DateStyle** property is equivalent to choosing the Master Items command in the File menu and choosing a style for the Date master item.

## Related Topics

### Language Elements

[Date Property](#)

[DateShown Property](#)

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[UpdateDateAndTime Method](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

**Related**

## DateStyle Property Example

This example uses the **DateStyle** property of the MasterItems object to find and report the date style for the Date master item.

```
Dim ABC As Object, Chart As Object, MasterItems As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                             ' Get the active chart
Set MasterItems = Chart.MasterItems

Dim Date_Style As Single
Date_Style = MasterItems.DateStyle                     ' Find date style

Select Case Date_Style                                 ' Report date style
    Case 0
        MsgBox "Date Style is MM/DD/YY."
    Case 1
        MsgBox "Date Style is <short text>."
    Case 2
        MsgBox "Date Style is <long text>."
End Select
```

## LogoPathname Property

Related

Related

Related

### Usage

*MasterItemsObject*.**LogoPathname** = *PathName*

### Description

You use the **LogoPathname** property to find or set the pathname of the Logo master item. The **LogoPathname** property is read/write.

### Data Type

String

### Value

The pathname of the Logo master item

### ABC Equivalent

The **LogoPathname** property is equivalent to choosing Master Items in the File menu, clicking the Logo button in the Master Items dialog box, and choosing a file to serve as the logo.

## Related Topics

### Language Elements

[HideAll Method](#)

[Logo Property](#)

[LogoShown Property](#)

[Range Property](#)

[ShowAll Method](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Related

### LogoPathname, Range Properties Example

This example uses the **LogoPathname** property and **Range** property of the MasterItems object to find and report the date style for the master item date. The example assumes that there is a logo selection in the chart.

```
Dim ABC As Object, Chart As Object, MasterItems As Object
Dim Logo_Path_Name As String
Dim Range As Single

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Logo_Path_Name = MasterItems.LogoPathname            ' Get path to master items logo

MsgBox Logo_Path_Name                                  ' Display path to master items logo

Range = MasterItems.Range                             ' Get master item page range
Select Case Range
    Case 0
        MsgBox "Master items are only on first page."
    Case 1
        MsgBox "Master items are on all pages."
End Select
```

## Range Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject.Range = RangeIndicator</i>
<b>Description</b>	You use the <b>Range</b> property to find or set the range of pages that display the master items. The <b>Range</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The range of pages on which master items are shown, using the values in the following table.
	<b><i>RangeIndicator</i></b> <b>Pages</b>
	0 First page only
	1 All pages
<b>ABC Equivalent</b>	The <b>Range</b> method is equivalent to choosing Master Items in the File menu and choosing a Range option.

## **Related Topics**

### **Language Elements**

[HideAll Method](#)

[ShowAll Method](#)

### **Description**

[Displaying Master Items](#)

### **Object**

[MasterItems Object](#)

## ChartNameShown Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject</i> . <b>ChartNameShown</b> = {True   False}
<b>Description</b>	You use the <b>ChartNameShown</b> property to find or set whether the ChartName master item is displayed. The <b>ChartNameShown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the chart name master item; False does not.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[ChartName Property](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Related

### ChartNameShown, ChartName, MasterItems Properties Example

This example uses the **ChartNameShown** property and **ChartName** property of the MasterItems object and the **MasterItems** property of the Chart object to determine if the chart name master item is shown.

```
Dim ABC As Object, Chart As Object
Dim Master_Items As Object
Dim Chart_Name_Visible As Integer           ' For ChartNameShown property value

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True                          ' Make ABC visible
Set Master_Items = Chart.MasterItems

Chart_Name_Visible = Master_Items.ChartNameShown ' Get ChartNameShown property value

Select Case Chart_Name_Visible              ' Display return results
    Case True
        MsgBox "Chart name is visible. It is " + Master_Items.ChartName.Text + "."
    Case Else
        MsgBox "Chart name is not visible."
End Select
```

## DateShown Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject.DateShown</i> = {True   False}
<b>Description</b>	You use the <b>DateShown</b> property to find or set whether the Date master item is displayed. The <b>DateShown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the date master item; False does not.
<b>ABC Equivalent</b>	The <b>DateShown</b> property is equivalent to choosing the Master Items command in the File menu and selecting or deselecting the Date option.

## Related Topics

### Language Elements

[Date Property](#)

[DateStyle Property](#)

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[UpdateDateAndTime Method](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

## Related

### DateShown, Date Properties Example

This example uses the **DateShown** property and the **Date** property of the MasterItems object to determine if the Date master item is shown and display it if it is.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Date_Visible As Integer           ' For DateShown property return value

Set ABC = CreateObject("ABCFlow.application")   ' Start ABC
ABC.Visible = True                             ' Make ABC visible
Set Master_Items = Chart.MasterItems

Date_Visible = Master_Items.DateShown         ' Get DateShown property

Select Case Date_Visible                   ' Display return results
    Case True
        MsgBox "Date is visible. It is " + Master_Items.Date.Text + "."
    Case Else
        MsgBox "Date is not visible."
End Select
```

## LogoShown Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject</i> . <b>LogoShown</b> = {True   False}
<b>Description</b>	You use the <b>LogoShown</b> property to find or set whether the Logo master item is displayed. The <b>LogoShown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the logo; False does not.
<b>ABC Equivalent</b>	The <b>LogoShown</b> property is equivalent to choosing Master Items in the File menu and selecting or deselecting the Show Logo option in the Master Items dialog box.

## **Related Topics**

### **Language Elements**

[HideAll Method](#)

[Logo Property](#)

[LogoPathname Property](#)

[Range Property](#)

[ShowAll Method](#)

### **Description**

[Displaying Master Items](#)

### **Object**

[MasterItems Object](#)

## Related

### LogoShown, Logo Properties Example

This example uses the **LogoShown** property and the **Logo** property of the **MasterItems** object to determine if the logo master item is shown and display its width. For the width of the logo to be shown, there must be a logo in the chart.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Logo_Visible As Integer           ' For LogoShown property return value

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Master_Items = Chart.MasterItems

Logo_Visible = Master_Items.LogoShown                 ' Get LogoShown property

Select Case Logo_Visible                             ' Display return results
    Case True
        MsgBox "Logo is visible. Its width is " + Master_Items.Logo.Width + "."
    Case Else
        MsgBox "Logo is not visible."
End Select
```

## PageNumberShown Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject</i> . <b>PageNumberShown</b> = {True   False}
<b>Description</b>	You use the <b>PageNumberShown</b> property to find or set whether the PageNumber master item is displayed. The <b>PageNumberShown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the page number master item; False does not.
<b>ABC Equivalent</b>	The <b>PageNumberShown</b> property is equivalent to choosing the Master Items command in the File menu and selecting or deselecting the Page Numbers option.

## **Related Topics**

### **Language Elements**

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

### **Description**

[Displaying Master Items](#)

### **Object**

[MasterItems Object](#)

**Related**

## PageNumberShown, PageNumber Properties Example

This example uses the **PageNumberShown** property and the **PageNumber** property of the MasterItems object to determine if the Page number master item is shown. It then shows the text of the page number.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Page_Number_Visible As Integer           ' For PageNumberShown property value

Set ABC = CreateObject("ABCFlow.application") ' Start ABC
ABC.Visible = True                          ' Make ABC visible
Set Master_Items = Chart.MasterItems

Page_Number_Visible = Master_Items.PageNumberShown ' Get PageNumberShown property

Select Case Page_Number_Visible             ' Display return results
    Case True
        MsgBox "Page Number visible. Format is " + Master_Items.PageNumber.Text + "."
    Case Else
        MsgBox "Page Number is not visible."
End Select
```

## TimeShown Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject</i> . <b>TimeShown</b> = {True   False}
<b>Description</b>	You use the <b>TimeShown</b> property to find or set whether the Time master item is shown. The <b>TimeShown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the Time master item; False does not show it.
<b>ABC Equivalent</b>	The <b>TimeShown</b> property is equivalent to choosing Master Items in the File menu and selecting or deselecting the Show Time option.

## **Related Topics**

### **Language Elements**

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[Time Property](#)

[UpdateDateAndTime Method](#)

### **Description**

[Displaying Master Items](#)

### **Object**

[MasterItems Object](#)

**Related**

## TimeShown, Time Properties Example

This example uses the **TimeShown** property and the **Time** property of the MasterItems object to determine if the time master item is shown. If it is shown, the program gives its value.

```
Dim ABC As Object
Dim Chart As Object
Dim Master_Items As Object
Dim Time_Visible As Integer           ' For TimeShown property return value

Set ABC = CreateObject("ABCFlow.application")   ' Start ABC
ABC.Visible = True                             ' Make ABC visible
Set Master_Items = Chart.MasterItems

Time_Visible = Master_Items.TimeShown          ' Get TimeShown property

Select Case Time_Visible                    ' Display result
    Case True
        MsgBox "Time is visible. It is " + Master_Items.Time.Text + "."
    Case Else
        MsgBox "Time is not visible."
End Select
```

## Text1Shown Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject.Text1Shown</i> = {True   False}
<b>Description</b>	You use the <b>Text1Shown</b> property to find or set whether the Text1 master item is displayed. The <b>Text1Shown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the Text1 master item; False does not show it.
<b>ABC Equivalent</b>	The <b>Text1Shown</b> property is equivalent to choosing Master Items in the File menu and entering text in the Text1 text box.

## Related Topics

### Language Elements

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[Text1 Property](#)

[Text2 Property](#)

[Text2Shown Property](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)



## Text2Shown Property

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject.Text2Shown</i> = {True   False}
<b>Description</b>	You use the <b>Text2Shown</b> property to find or set whether the Text2 master item is displayed. The <b>Text2Shown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True shows the Text2 master item; False does not show it.
<b>ABC Equivalent</b>	The <b>Text2Shown</b> property is equivalent to choosing Master Items in the File menu and entering text in the Text2 text box.

## **Related Topics**

### **Language Elements**

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[Text1 Property](#)

[Text2 Property](#)

### **Description**

[Displaying Master Items](#)

### **Object**

[MasterItems Object](#)

## HideAll Method

Related

Related

Related

**Usage** *MasterItemsObject.HideAll*

**Description** You use the **HideAll** method to hide the master items in the chart.

**ABC Equivalent** The **HideAll** method is equivalent to choosing the Master Items command in the File menu and deselecting all the master items options.

## Related Topics

### Language Elements

[ChartNameShown Property](#)  
[DateShown Property](#)  
[LogoShown Property](#)  
[PageNumberShown Property](#)  
[Range Property](#)  
[ShowAll Method](#)  
[Text1Shown Property](#)  
[Text2Shown Property](#)  
[TimeShown Property](#)  
[UpdateDateAndTime Method](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)



## ShowAll Method

Related

Related

Related

**Usage** *MasterItemsObject.ShowAll*

**Description** You use the **ShowAll** method to display the master items in the chart.

**ABC Equivalent** The **ShowAll** method is equivalent to choosing the Master Items command in the File menu and selecting all the master items options.

## Related Topics

### Language Elements

[ChartNameShown Property](#)  
[DateShown Property](#)  
[HideAll Method](#)  
[LogoShown Property](#)  
[PageNumberShown Property](#)  
[Range Property](#)  
[Text1Shown Property](#)  
[Text2Shown Property](#)  
[TimeShown Property](#)  
[UpdateDateAndTime Method](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)



## UpdateDateAndTime Method

Related

Related

Related

<b>Usage</b>	<i>MasterItemsObject.UpdateDateAndTime</i> [ <i>Date</i> ] [, <i>Time</i> ] The <i>Date</i> element, which is optional, specifies a specific date. The <i>Time</i> element, which is optional, specifies a specific time.
<b>Description</b>	You use the <b>UpdateDateAndTime</b> method to update the master item time and date. If you omit the elements, the data and time are changed to the system date and time. You can optionally supply a date and a time.
<b>Data Type</b>	The <i>Date</i> element and <i>Time</i> element are strings.
<b>Value</b>	None
<b>ABC Equivalent</b>	The <b>UpdateDateAndTime</b> method is equivalent to choosing Master Items in the File menu and clicking the Update Date and Time button.

## Related Topics

### Language Elements

[Date Property](#)

[DateShown Property](#)

[DateStyle Property](#)

[HideAll Method](#)

[Range Property](#)

[ShowAll Method](#)

[Time Property](#)

[TimeShown Property](#)

### Description

[Displaying Master Items](#)

### Object

[MasterItems Object](#)

**Related**

## UpdateDateAndTime Method Example

This example uses the **UpdateDateAndTime** method of the MasterItems object to update the date and time master items to the current system date and time.

```
Dim ABC As Object
Dim Chart As Object
Dim MasterItems As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set MasterItems = Chart.MasterItems

MasterItems.UpdateDateAndTime                          ' Update current Master Items date and time
settings

MsgBox "You've just updated your Master Item time and date settings."
```

## Menu Collection

**Description** The Menu collection is below the Application object. Below the Menu collection are the MenuItem objects. You can have multiple MenuItem objects in the Menu collection.

### Properties

[Application](#)  
[Count](#)  
[Parent](#)  
[Text](#)  
[Visible](#)

### Methods

[AppendItem](#)  
[DeleteItem](#)  
[DeleteAll](#)  
[InsertItem](#)  
[Item](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

[VBX Event Variables](#)

## Text Property (Menu Collection)

Related

Related

Related

**Usage** *MenuCollection.Text = MenuName*

**Description** The **Text** property of the Menu collection lets you change the name of a menu after you have created it. You may include the "&" character for keyboard shortcuts. The **Text** property is read/write.

**Data Type** String

**Value** The text of the menu

**ABC Equivalent** None

## Related Topics

### Language Elements

[AddMenu Method](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[RemoveMenu Method](#)

[Text Property \(MenuItem Object\)](#)

[Text Property \(Object Object\)](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Adding Menus](#)

### Object

[Menu Collection](#)

**Related**

## Text Property (Menu Collection) Example

This example uses the **Text** property of the Menu collection to change the name of a menu after it is created.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible

Set Menu = ABC.AddMenu("Statistics", ABC1.VBX, Form1.Caption) ' Add a new menu item

ABC.MsgBox "Click OK to see the menu text change."

Menu.Text = "Organization"                        ' Change the new menu's text
```

## Visible Property (Menu Collection)

Related

Related

Related

<b>Usage</b>	<i>MenuCollection.Visible</i> = {True   False}
<b>Description</b>	The <b>Visible</b> property of the Menu collection lets you show or hide a menu. The <b>Visible</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True makes the menu visible; False hides it.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[AddMenu Method](#)

[Checked Property](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[Enabled Property](#)

[RemoveMenu Method](#)

[Text Property \(MenuItem Object\)](#)

[Visible property \(Application object\)](#)

### Description

[Adding Menus](#)

### Object

[Menu Collection](#)

**Related**

## Visible Property (Menu Collection) Example

This example uses the **Visible** property of the Menu collection to hide and reshow a menu.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                               ' Make ABC visible

Set Menu = ABC.AddMenu("Statistics", ABC1.VBX, Form1.Caption) ' Create a new menu

ABC.MsgBox "Click OK to hide the new menu."

Menu.Visible = False                             ' Hide the new menu

ABC.MsgBox "Click OK to see the new menu reappear"

Menu.Visible = True                              ' Unhide the new menu
```

## AppendItem Method

Related

Related

Related

<b>Usage</b>	<i>MenuCollection.AppendItem (ItemName)</i> The <i>ItemName</i> element is the name of the item you wish to add to the menu.
<b>Description</b>	The <b>AppendItem</b> method lets you add a menu item to the next position, below any existing items, in a menu you created. It is customary to list items within groups in alphabetical order. If you use the name of an existing menu, the method returns the existing MenuItem object. Otherwise it returns the new MenuItem object.
<b>Data Type</b>	Object
<b>Value</b>	The menu item you created
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[AddMenu Method](#)

[Checked Property](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[Enabled Property](#)

[InsertItem Method](#)

[Item Method \(Charts Collection\)](#)

[Item Method \(FieldTemplates Collection\)](#)

[Item Method \(FieldValues Collection\)](#)

[Item Method \(Objects Collection\)](#)

[RemoveMenu Method](#)

### Description

[Adding Menus](#)

### Object

[Menu Collection](#)

## Deleteltem Method

Related

Related

Related

**Usage** *MenuCollection.Deleteltem MenuName*

The *MenuName* element is the name of the menu item.

**Description** The **Deleteltem** method lets you delete a menu item from a menu.

**ABC Equivalent** None

## Related Topics

### Language Elements

[AppendItem Method](#)

[Checked Property](#)

[DeleteAll Method](#)

[InsertItem Method](#)

[Item Method \(Menu Collection\)](#)

[Text Property \(MenuItem Object\)](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Adding Menus](#)

### Object

[Menu Collection](#)

**Related**

## DeleteItem Method Example

This example uses the **DeleteItem** method of the Menu collection to delete a menu item.

```
Dim ABC As Object, Menu As Object, MenuItem As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

Set Menu = ABC.AddMenu("Test", ABC1.VBX, Form1.Caption) ' Create the main menu

Set MenuItem = Menu.AppendItem("First Item")           ' Add items to the new menu
Menu.AppendItem ("Second Item")

MsgBox "Click on the ABC application to see the new menu items."

MsgBox "Click OK to delete a menu item."

Menu.DeleteItem MenuItem                               ' Delete a menu item
```

## DeleteAll Method

Related

Related

Related

**Usage** *MenuCollection.DeleteAll*

**Description** The **DeleteAll** method lets you remove all items from a menu.

**ABC Equivalent** None

## Related Topics

### Language Elements

[AddMenu Method](#)

[Checked Property](#)

[DeleteItem Method](#)

[RemoveMenu Method](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Adding Menus](#)

### Object

[Menu Collection](#)



## InsertItem Method

Related

Related

Related

<b>Usage</b>	<i>MenuCollection.InsertItem</i> <i>ItemName</i> , ({ <i>PreviousItem</i>   <i>Position</i> }) The <i>ItemName</i> element is the name of the item you wish to add to the menu. The <i>PreviousItem</i> element is the item to position the new item after. The <i>Position</i> element is the numeric position of the new item.
<b>Description</b>	The <b>InsertItem</b> method lets you insert a menu item in a specified position in a menu you created. It is customary to list items within groups in alphabetical order. You provide the title of the item you wish to create, followed by the position of the item, specified either by giving the name of the existing item that the new item should be placed after or by specifying the numerical position of the item.
<b>Data Type</b>	Object
<b>Value</b>	The menu item you created
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[AddMenu Method](#)

[AppendItem Method](#)

[Checked Property](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[Enabled Property](#)

[Item Method \(Menu Collection\)](#)

[RemoveMenu Method](#)

### Description

[Adding Menus](#)

### Object

[Menu Collection](#)

## Related

### InsertItem Method Example

This example uses the **InsertItem** method of the Menu collection to insert a menu item between two existing items.

```
Dim ABC As Object, Menu As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible

Set Menu = ABC.AddMenu("Test", ABC1.VBX, Form1.Caption) ' Create a new menu

Menu.AppendItem ("First Item")                    ' Create first menu item
Menu.AppendItem ("Second Item")                  ' Create second menu item

Menu.InsertItem ("Third Item", 2)                 ' Insert third item between first two
```

## Item Method (Menu Collection)

Related

Related

Related

<b>Usage</b>	<i>MenuCollection.Item</i> ({ <i>ItemText</i>   <i>Position</i> }) The <i>ItemText</i> element is the text of the item you want to find. The <i>Position</i> element is the numeric position of the item in the menu.
<b>Description</b>	The <b>Item</b> method of the Menu collection lets you find a menu item either by its text or by its location in a menu.
<b>Data Type</b>	Object
<b>Value</b>	A menu item
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[AddMenu Method](#)

[AppendItem Method](#)

[Checked Property](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[Enabled Property](#)

[InsertItem Method](#)

[RemoveMenu Method](#)

[Text Property \(MenuItem Object\)](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Adding Menus](#)

### Object

[Menu Collection](#)



## MenuItem Object

**Description** The MenuItem object is below the Menu collection. You can have multiple MenuItem objects.

### Properties

---

[Application](#)  
[Checked](#)  
[Enabled](#)  
[Parent](#)  
[Text](#)

### Methods

---

There are no methods for the MenuItem object.

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

[VBX Event Variables](#)

## Checked Property

Related

Related

Related

**Usage** *MenuItemObject.Checked* = {True | False}

**Description** The **Checked** property lets you show or hide a check mark beside a menu item. The **Checked** property is read/write.

**Data Type** Integer (Boolean)

**Value** True shows a check mark beside the item; False hides it.

**ABC Equivalent** None

## Related Topics

### Language Elements

[AddMenu Method](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[Enabled Property](#)

[RemoveMenu Method](#)

[Text Property \(MenuItem Object\)](#)

### Description

[Adding Menus](#)

### Object

[MenuItem Object](#)



## Enabled Property

Related

Related

Related

**Usage** *MenuItemObject.Enabled* = {True | False}  
**Description** The **Enabled** property lets you show a menu item or make it gray. The **Enabled** property is read/write.  
**Data Type** Integer (Boolean)  
**Value** True enables the item; False grays it.  
**ABC Equivalent** None

## Related Topics

### Language Elements

[AddMenu Method](#)

[Checked Property](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[RemoveMenu Method](#)

[Text Property \(MenuItem Object\)](#)

### Description

[Adding Menus](#)

### Object

[MenuItem Object](#)



## Text Property (MenuItem Object)

Related

Related

Related

<b>Usage</b>	<i>MenuItemObject.Text = ItemName</i>
<b>Description</b>	The <b>Text</b> property of the MenuItem object lets you change the name of a menu item after you have added it to a menu. You may include the "&" character for keyboard shortcuts. The <b>Text</b> property is read/write.
<b>Data Type</b>	String
<b>Value</b>	The text of the menu item
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[AppendItem Method](#)

[Checked Property](#)

[DeleteAll Method](#)

[DeleteItem Method](#)

[InsertItem Method](#)

[Text Property \(Menu Collection\)](#)

[Text Property \(Object Object\)](#)

[Visible Property \(Menu Collection\)](#)

### Description

[Adding Menus](#)

### Object

[MenuItem Object](#)



## Object Object

**Description** The Object object is contained in the Object collection. You can have multiple Object objects in the collection. Below the Object object are the Shape, Line\_, TextBlock, OLE, Font, and FieldValues objects. Each Object object can have multiple FieldValue objects, but only one Shape, Line\_, TextBlock, OLE, and Font object for each Object object. Note that the Shape object and Line\_ objects are mutually exclusive. If the Object object is a shape, the Line\_ object is a meaningless placeholder.

### Properties

---

[Application](#)  
[Bottom](#)  
[CenterX](#)  
[CenterY](#)  
[Color](#)  
[FieldValues](#)  
[Font](#)  
[Height](#)  
[Left](#)  
[Line\\_](#)  
[OLE](#)  
[Parent](#)  
[Right](#)  
[Selected](#)  
[Shape](#)  
[StretchType](#)  
[Text](#)  
[TextAlignment](#)  
[TextBlock](#)  
[TextLF](#)  
[Top](#)  
[Type](#)  
[UniqueID](#)  
[Valid](#)  
[Width](#)

### Methods

---

[Clear\\_](#)  
[Duplicate](#)  
[Repaint](#)  
[RestorePicture](#)  
[ToBack](#)  
[ToFront](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

[VBX Event Variables](#)

## Application Property

Related

Related

Related

**Usage** *Object***Object.Application**

**Description** You use the **Application** property to find the running Application object. You can find the object to which the master items apply and which master items to display. The **Application** property is read only.

**Data Type** Object

**Value** The running application

**ABC Equivalent** None

## Related Topics

### Language Elements

[ChartName Property](#)

[ChartNameShown Property](#)

[Parent Property](#)

### Description

[Displaying Master Items](#)

### Object

[Application Object](#)

[Chart Object](#)

[Charts Collection](#)

[FieldTemplate Object](#)

[FieldTemplates Collection](#)

[FieldValue Object](#)

[FieldValues Collection](#)

[Font Object](#)

[Line Object](#)

[MasterItems Object](#)

[Menu Collection](#)

[MenuItem Object](#)

[Object Object](#)

[Objects Collection](#)

[OLE Object](#)

[PageLayout Object](#)

[Preferences Object](#)

[Shape Object](#)

[TextBlock Object](#)

## Related

### Application Property Example

This example finds the **Application** property value of the Object object. The other **Application** property values are found in similar ways.

```
Dim ABC As Object
Dim Objects_Parent As Object
Dim Chart As Object
Dim Objects As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

Set Chart = ABC.ActiveChart
Set Objects = Chart.Objects

Set Objects_Parent = Objects.Application               ' Set the Objects Object
MsgBox "The running application is " + Objects_Parent
```

## Bottom Property (Object Object)

Related

Related

Related

### Usage

*ObjectObject*.**Bottom** = *Distance*

### Description

The **Bottom** property lets you find or set the location of the bottom of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units for measuring the distance using the **Units** property. The **Bottom** property is read/write.

### Data Type

Double

### Value

The location of the bottom of the object

### ABC Equivalent

None

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[CenterX Property](#)

[CenterY Property](#)

[Left Property \(Object Object\)](#)

[Right Property \(Object Object\)](#)

[Top Property \(Object Object\)](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Moving Objects](#)

### Object

[Object Object](#)

## CenterX Property

Related

Related

Related

<b>Usage</b>	<i>ObjectObject.CenterX = Distance</i>
<b>Description</b>	The <b>CenterX</b> property lets you find or set the center of the object. The property does not affect the size of the object. You set the units used to measure the distance using the <b>Units</b> property. The <b>CenterX</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The center of the object
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[CenterY Property](#)

[Left Property \(Object Object\)](#)

[Right Property \(Object Object\)](#)

[Top Property \(Object Object\)](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Moving Objects](#)

### Object

[Object Object](#)

## CenterY Property

Related

Related

Related

<b>Usage</b>	<i>ObjectObject.CenterY = Distance</i>
<b>Description</b>	The <b>CenterY</b> property lets you find or set the center of the object. The property does not affect the size of the object. You set the units used to measure the distance using the <b>Units</b> property. The <b>CenterY</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The center of the object
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[CenterX Property](#)

[Left Property \(Object Object\)](#)

[Right Property \(Object Object\)](#)

[Top Property \(Object Object\)](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Moving Objects](#)

### Object

[Object Object](#)

## Color Property (Object Object)

Related

Related

Related

### Usage

*ObjectObject.Color = Color*

### Description

The **Color** property of the Object object lets you find or set the fill color for shapes, the color of lines, or the color of text (see the **MakeRGB** method). Using the **Color** property for shapes is the same as using the **FillColor** property. Using the **Color** property for lines finds the stem color and sets the color of the stem and both ends. The **Color** property is read/write.

### Data Type

Long

### Value

The color for a shape, line, or text object

### ABC Equivalent

The **Color** property of the Object object is equivalent to selecting a shape, clicking the Apply button next to the Fill Pattern button in the Shape ribbon, and clicking the color you want.

## Related Topics

### Language Elements

[BasicColor Method](#)

[BorderColor Property](#)

[Color Property \(Font Object\)](#)

[Color Property \(Line Object\)](#)

[FillColor Property](#)

[MakeRGB Method](#)

[ShadowColor Property](#)

### Description

[Setting Shape Colors](#)

[Setting Line Colors](#)

[Setting Text Colors](#)

[Formatting Shape Numbers](#)

[Fill, Border, and Shadow Colors](#)

[Text Color](#)

### Object

[Object Object](#)

## Related

# Color Property, Height Property, Width Property (Object Object), and FillColor Property Example

This example uses the **Color** property, the **Height** property, and the **Width** property of the Object object and the **FillColor** property of the Shape object to set the color, width, and height of shapes.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim NewShapel As Object, NewShape2 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set NewShapel = Chart.DrawShape("Decision")
NewShapel.Color = ABC.RED
NewShapel.Height = 1
NewShapel.Width = 2

Set NewShape2 = Chart.DrawShape("Operation")
NewShape2.Shape.FillColor = ABC.MakeRGB(0, 0, 255)
NewShape2.Height = .5
NewShape2.Width = 1
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Draw a Decision shape  
' Make the shape red  
' Make the shape 1 inch high  
' Make the shape 2 inches wide  
  
' Draw an Operation shape  
' Make the shape blue  
' Make the shape 1/2 inch high  
' Make the shape 1 inch wide

## FieldValues Property

Related

Related

Related

### Usage

*ObjectObject*.FieldValues

### Description

The **FieldValues** property lets you find the data fields included in the FieldValues collection. The **FieldValues** property is read only, but all the properties from the object it returns are read/write.

### Data Type

Collection object

### Value

The fields included in the FieldValues collection

### ABC Equivalent

None

## Related Topics

### Description

[Working with Data Field Values](#)

### Object

[Object Object](#)

## Related

### FieldValues Property Example

This example uses the **FieldValues** property of the Object object to enter text in a data field.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, Shapel As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Client")         ' Create a field
Field1.Type = 0                                        ' Make the field's type text
Set Shapel = Chart.DrawShape                           ' Draw a shape
Shapel.FieldValues.Item("Client").Value = "John P. Cliché" ' Enter text in the field
```

## Font Property

Related

Related

Related

<b>Usage</b>	<i>Object</i> <b>Object.Font</b>
<b>Description</b>	The <b>Font</b> property lets you find the font object for text. The <b>Font</b> property is read only, but all the properties from the object it returns are read/write.
<b>Data Type</b>	Object
<b>Value</b>	The Font object for text
<b>ABC Equivalent</b>	None

## Related Topics

### Description

[Text Typeface and Size](#)

### Object

[Object Object](#)

**Related**

## Font Property Example

This example uses the **Font** property of the Object object to change text attributes.

```
Dim ABC As Object, Chart As Object
Dim Text1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Text1 = Chart.DrawTextBlock("OLE Automation is fun!")  ' Create a line of text
Text1.Font.Italic = True                            ' Make the text italic
```

## Height Property (Object Object)

Related

Related

Related

**Usage** *ObjectObject.Height = Height*

**Description** The **Height** property lets you find or set the height of the object. You set the units used to measure the height using the **Units** property. The **Height** property is read/write.

**Data Type** Double

**Value** The height of the object

**ABC Equivalent** None

## Related Topics

### Language Elements

- [Height Property \(Application Object\)](#)
- [Height Property \(PageLayout Object\)](#)
- [StretchType Property](#)
- [Units Property \(Chart Object\)](#)
- [Units Property \(Preferences Object\)](#)
- [Width Property \(Object Object\)](#)

### Description

- [Resizing Objects](#)

### Object

- [Object Object](#)

## Left Property (Object Object)

Related

Related

Related

### Usage

*ObjectObject.Left = Distance*

### Description

The **Left** property lets you find or set the location of the left side of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units for measuring the distance using the **Units** property. The **Left** property is read/write.

### Data Type

Double

### Value

The location of the left side of the object

### ABC Equivalent

None

## Related Topics

### Language Elements

[Bottom Property \(Object Object\)](#)

[CenterX Property](#)

[CenterY Property](#)

[Left Property \(Application Object\)](#)

[Right Property \(Object Object\)](#)

[Top Property \(Object Object\)](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Moving Objects](#)

### Object

[Object Object](#)

## Line\_Property

Related

Related

Related

<b>Usage</b>	<i>Object</i> <i>Object</i> . <b>Line_</b>
<b>Description</b>	The <b>Line_</b> property lets you find the line objects. The <b>Line_</b> property is read only, but all the properties from the object it returns are read/write.
<b>Data Type</b>	Object
<b>Value</b>	The Line_ object
<b>ABC Equivalent</b>	None

## Related Topics

### Description

[Drawing Lines](#)

### Object

[Object Object](#)

**Related**

## Line Property Example

This example uses the **Line\_** property of the Object object to change the style of a line stem.

```
Dim ABC As Object, Chart As Object
Dim Line1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Line1 = Chart.DrawFreeLine(5, 2)
Line1.Line_.StemStyle = 4
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Draw a plain line  
' Change the stem style

## Parent Property

Related

Related

Related

<b>Usage</b>	<i>Object.Parent</i>
<b>Description</b>	You use the <b>Parent</b> property to find the parent object of an object. For example, the parent of the Application object is the running ABC application. The parent of the Objects collection is the chart object in which the objects reside. The <b>Parent</b> property is read only.
<b>Data Type</b>	Object
<b>Value</b>	The parent of the object
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Application Property](#)

### Description

[Adjusting the Page Layout](#)

### Object

[Application Object](#)

[Chart Object](#)

[Charts Collection](#)

[FieldTemplate Object](#)

[FieldTemplates Collection](#)

[FieldValue Object](#)

[FieldValues Collection](#)

[Font Object](#)

[Line Object](#)

[MasterItems Object](#)

[Menu Collection](#)

[MenuItem Object](#)

[Object Object](#)

[Objects Collection](#)

[OLE Object](#)

[PageLayout Object](#)

[Preferences Object](#)

[Shape Object](#)

[TextBlock Object](#)

**Related**

## Parent Property Example

This example uses the **Parent** property of the Application object to put the parent of the application into a variable. The **Parent** properties of the other objects and collections work the same way.

```
Dim ABC As Object
Dim App_Parent As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible

Set App_Parent = ABC.Parent                             ' Set the collection of open ABC charts
```

## Right Property (Object Object)

Related

Related

Related

### Usage

*ObjectObject.Right = Distance*

### Description

The **Right** property lets you find or set the location of the right side of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units used for measuring the distance using the **Units** property. The **Right** property is read/write.

### Data Type

Double

### Value

The location of the right side of the object

### ABC Equivalent

None

## Related Topics

### Language Elements

[Bottom Property \(Application Object\)](#)

[CenterX Property](#)

[CenterY Property](#)

[Left Property \(Object Object\)](#)

[Right Property \(Application Object\)](#)

[Top Property \(Object Object\)](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Moving Objects](#)

### Object

[Object Object](#)

## StretchType Property

Related

Related

Related

### Usage

*ObjectObject*.**StretchType** = *Value*

### Description

The **StretchType** property lets you find or set the type of stretching for the object. You can set it to normal (opposite sides both move as when you stretch in ABC) or so that one side is fixed. If the user stretches with one side fixed, it is the same as if he or she pressed **Ctrl** while stretching. If you resize using ABC OLE Automation, then the top and left sides are held fixed as if you were stretching from the right or bottom center handle and holding the **Ctrl** key. The **StretchType** property is read/write.

### Data Type

Integer

### Value

The values for the stretch types are in the following table.

<b>Value</b>	<b>Meaning</b>
0	Normal
1	Fixed sides

**ABC Equivalent** None

## Related Topics

### Language Elements

[Height Property \(Object Object\)](#)

[Width Property \(Object Object\)](#)

### Description

[Resizing Objects](#)

### Object

[Object Object](#)

## Related

# StretchType Property and DrawDirection Property Example

This example uses the **StretchType** property of the Object object and the **DrawDirection** property of the Chart object to set the type of stretching for an object.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Chart.DrawDirection = 2
Set Shape1 = Chart.DrawShape
Set Shape2 = Chart.DrawShape

Shape1.StretchType = 0
Shape1.Text = "Normal"
Shape1.Color = ABC.Cyan
MsgBox "You can resize the cyan object normally."
Shape2.StretchType = 1
Shape2.Text = "Opposite Side Fixed"
Shape2.Color = ABC.Yellow
MsgBox "You can resize each side of the yellow object independently."
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' Draw new shapes down the page
' Draw the first shape
' Draw the second shape

' Use Normal stretch type
' Enter text in the shape
' Apply a color to the shape

' Use OppositeSideFixed stretch type
' Enter text in the shape
' Apply a color to the shape
```

## Selected Property

Related

Related

Related

**Usage** *ObjectObject.Selected* = {True | False}

**Description** You use the **Selected** property to find or set whether an object is selected. The **Selected** property is read/write.

**Data Type** Integer (Boolean)

**Value** True means the object is selected; False means the object is not selected.

**ABC Equivalent** The **Selected** property is equivalent to clicking an object to select it.

## Related Topics

### Language Elements

[DeselectAll Method](#)

[Select Method](#)

[SelectShape Type Method](#)

### Description

[Selecting Objects in a Chart](#)

[Selecting Shapes](#)

### Object

[Object Object](#)



## Shape Property

Related

Related

Related

<b>Usage</b>	<i>Object</i> <b>Object.Shape</b>
<b>Description</b>	You use the <b>Shape</b> property to find the shape object. The <b>Shape</b> property is read only, but the properties from the object it returns are read/write.
<b>Data Type</b>	Object
<b>Value</b>	The Shape object
<b>ABC Equivalent</b>	None

## Related Topics

### Description

[Working with Shapes](#)

### Object

[Object Object](#)

**Related**

## Shape Property Example

This example uses the **Shape** property of the Object object to set the color of a shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Shape1 = Chart.DrawShape
Shape1.Shape.FillColor = ABC.MakeRGB(0, 127, 127)
FillColor
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Draw a shape  
' Access the Shape Object property

## Text Property (Object Object)

Related

Related

Related

<b>Usage</b>	<i>ObjectObject.Text = TextString</i>
<b>Description</b>	You use the <b>Text</b> property of the Object object to add or read text inside any shape or text block. If you wish to preserve Returns when reading the text, you should use the <b>TextLF</b> property. The <b>Text</b> property is read/write.
<b>Data Type</b>	String
<b>Value</b>	The text inside a shape
<b>ABC Equivalent</b>	The <b>Text</b> property of the Object object is equivalent to typing while a shape is selected.

## Related Topics

### Language Elements

[FitShapeToText Method](#)

[Text Property \(Menu Collection\)](#)

[Text Property \(MenuItem Object\)](#)

[TextLF Property](#)

### Description

[Adding Text to Shapes](#)

[Adding Text to a Shape](#)

### Object

[Object Object](#)

## Related

### Text Property (Object Object) Example

This example uses the **Text** property of the Object object to add text to a shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape("Document")               ' Draw a Document shape
Shape1.Text = "I love chocolate!"                      ' Add text to the shape
```

## TextAlignment Property

Related

Related

Related

### Usage

*ObjectObject.TextAlignment = AlignmentChoice*

### Description

You use the **TextAlignment** property to align the text inside shapes and in text blocks or to find the alignment. The **TextAlignment** property is read/write.

### Data Type

Integer

### Value

The **TextAlignment** property uses the following values to represent each square in the Alignment Grid in the Text ribbon.

0	1	2
3	4	5
6	7	8

### ABC Equivalent

None

## Related Topics

### Language Elements

[Bold Property](#)

[Color Property \(Font Object\)](#)

[Italic Property](#)

[Opaque Property](#)

[Size Property](#)

[Underline Property](#)

### Description

[Text Alignment](#)

### Object

[Object Object](#)



## Top Property (Object Object)

Related

Related

Related

### Usage

*ObjectObject.Top = Distance*

### Description

The **Top** property lets you find or set the location of the top of the object based on the top left of the chart, which is at (0,0). The property does not affect the size of the object. You set the units to measure the distance using the **Units** property. The **Top** property is read/write.

### Data Type

Double

### Value

The location of the top of the object

### ABC Equivalent

None

## Related Topics

### Language Elements

[Bottom Property \(Object Object\)](#)

[CenterX Property](#)

[CenterY Property](#)

[Left Property \(Object Object\)](#)

[Right Property \(Object Object\)](#)

[Top Property \(Application Object\)](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Moving Objects](#)

### Object

[Object Object](#)

## Type Property (Object Object)

Related

Related

Related

### Usage

*ObjectObject*. **Type**

### Description

The **Type** property lets you find the type of object. The **Type** property is read only.

### Data Type

Integer

### Value

The values for the types are in the following table.

Object Type	Description
0	Shape
1	Line
2	Text
3	Bitmap
4	OLE client object
5	Master

**ABC Equivalent** None

## Related Topics

### Language Elements

[ShapeName Property](#)

[Type Property \(Chart Object\)](#)

[Type Property \(FieldTemplate Object\)](#)

[Type Property \(FieldValue Object\)](#)

[Type Property \(Line Object\)](#)

[UniqueID Property](#)

### Description

[Identifying an Object](#)

### Object

[Object Object](#)

## UniqueID Property

Related

Related

Related

### Usage

*ObjectObject.UniqueID*

### Description

The **UniqueID** property lets you find the ID for an object. You can use the ID to choose an object in the Objects collection using, for example, the **ItemFromUniqueID** method. The identifier is unique for each object in each chart. If you wish, you could create a database containing the **UniqueID** property values for all the objects in a chart to make it easy to identify and act on each of them. A **UniqueID** is never reused in a chart even if you delete the object. The **UniqueID** property is read only.

### Data Type

Double

### Value

The unique ID of the object

### ABC Equivalent

None

## **Related Topics**

### **Language Elements**

[ItemFromUniqueID Method](#)

[ShapeName Property](#)

[Type \(Object object\)](#)

### **Description**

[Identifying an Object](#)

### **Object**

[Object Object](#)

**Related**

## UniqueID Property Example

This example uses the **UniqueID** property of the Object object to find the unique identifier for an object.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape                           ' Draw a shape
MsgBox "The shape's unique ID is " + Shape1.UniqueID + "." ' Display the shape's ID
```

## Valid Property

Related

Related

Related

<b>Usage</b>	<i>ChartObject.Valid</i> <i>ObjectObject.Valid</i>
<b>Description</b>	You use the <b>Valid</b> property in the While part of a Do While loop to check that the ItemFrom methods are returning valid objects. The <b>Valid</b> property is read only.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the object is valid; False means the object is not valid.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[ItemFromAll Method](#)  
[ItemFromAttachments Method](#)  
[ItemFromFieldValue Method](#)  
[ItemFromLines Method](#)  
[ItemFromNumber Method](#)  
[ItemFromSelection Method](#)  
[ItemFromShapes Method](#)  
[ItemFromText Method](#)  
[ItemFromUniqueID Method](#)  
[ResetSearch Method](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Object Object](#)  
[Chart Object](#)

**Related**

## Valid Property and Type Property (Chart Object) Example

This example uses the **Valid** property and the **Type** property of the Chart object to find valid charts and display their types. Also see the example included with the [ItemFromSelection method \(Objects collection\)](#).

```
Dim ABC As Object, Chart As Object
Dim Path1 As String
Dim File1 As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                               ' Make ABC visible
ABC.CloseAll                                     ' Close all open charts

Path1 = ABC.Path + "\SAMPLES\QUALIT.AF3"          ' Path of file to be opened
Set File1 = ABC.Open(Path1)                       ' Open chart
Set Chart = ABC.ActiveChart                       ' Get the active chart

If Chart.Valid Then                               ' If the current chart is valid
    Chart.Minimize                                ' minimize its window and
    MsgBox Path1 + " is a " + Chart.Type + " type of chart." ' post message with type
    ABC.MsgBox "The minimized chart is a valid chart."
Else
    MSG1 = " was not found. Please enter a valid sample file name in the code and try again."
    MsgBox (Path1 + MSG1)
End If
```

## Width Property (Object Object)

Related

Related

Related

<b>Usage</b>	<i>ObjectObject.Width = Width</i>
<b>Description</b>	The <b>Width</b> property lets you find or set the width of the object. You set the units used to measure the width using the <b>Units</b> property. The <b>Width</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The width of the object
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Height Property \(Object Object\)](#)

[StretchType Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[Width Property \(Application Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Resizing Objects](#)

### Object

[Object Object](#)

## Duplicate Method (Object Object)

Related

Related

Related

**Usage** *Object**Object*.**Duplicate**

**Description** The **Duplicate** method of the Object object makes a duplicate of the selected object and returns the duplicate object.

**Data Type** Object

**Value** The duplicate object

**ABC Equivalent** None

## Related Topics

### Language Elements

[Copy Method](#)

[Duplicate Method \(Chart Object\)](#)

[Paste Method](#)

### Description

[Duplicating Objects](#)

[Speeding Actions](#)

### Object

[Object Object](#)

## Related

# Duplicate Method (Object Object), NumberShown Property, and Renumber Method Example

This example uses the **Duplicate** method of the Object object and the **NumberShown** property and **Renumber** method of the Shape object to duplicate a shape, show the number on the shape, and increment the number of the shape.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, NewShape As Object
Dim Count As Double

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Chart.NextNumber = 100
Set Shape1 = Chart.DrawShape("Terminal")               ' Draw a Terminal shape
For Count = 1 To 10                                    ' To make 10 copies of the shape
    Set NewShape = Shape1.Duplicate                    ' Duplicate the last shape
    NewShape.CenterY = Count / 2                       ' Move the new shape down
    NewShape.Shape.NumberShown = True                  ' Show the shape number
    NewShape.Shape.Renumber                             ' Increment the shape number
Next Count
```

## Repaint Method

Related

Related

Related

**Usage** *ChartObject.Repaint*  
*ObjectObject.Repaint*

**Description** You use the **Repaint** method to repaint the entire chart after a series of actions with the **NoRepaint** property set to True.

**ABC Equivalent** None

## Related Topics

### Language Elements

[NoRepaint Property](#)

### Description

[Speeding Actions](#)

### Object

[Chart Object](#)

[Object Object](#)



## RestorePicture Method (Object Object)

Related

Related

Related

### Usage

*ObjectObject*.**RestorePicture**

### Description

The **RestorePicture** method of the Object object lets you restore bitmap and OLE client objects to their original size. This method is nearly the same as the OLE object's **RestorePicture** method. The difference is that this method handles bitmaps as well as OLE objects, while the OLE object's **RestorePicture** method only works on OLE objects.

**ABC Equivalent** None

## Related Topics

### Language Elements

[RestorePicture Method \(OLE Object\)](#)

### Description

[Resizing Objects](#)

### Object

[Object Object](#)

## Related

### RestorePicture Method (Object Object) Example

This example uses the **RestorePicture** method of the Object object to restore bitmaps and OLE objects to their original sizes. For the program to have any effect, you must have a resized bitmap or OLE client object in the chart.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Everything = Chart.Objects
Do
    Set Current = Everything.ItemFromAll
    Current.RestorePicture
original size
Loop While Current.Valid

' Start ABC
' Make ABC visible
' Get the active chart

' Get all items in the chart

' Choose the next item
' Return bitmaps and OLE objects to their
```

## ToBack Method (Object Object)

Related

Related

Related

**Usage** *ObjectObject.ToBack*

**Description** You use the **ToBack** method of the Object object to move the object to the back.

**ABC Equivalent** The **ToBack** method is equivalent to clicking the Selection button and clicking the To Back button.

## Related Topics

### Language Elements

[ToBack Method \(Chart Object\)](#)

[ToFront Method \(Chart Object\)](#)

[ToFront Method \(Object Object\)](#)

### Description

[Changing the Display Order of Objects](#)

### Object

[Object Object](#)



## ToFront Method (Object Object)

Related

Related

Related

**Usage** *ObjectObject.ToFront*

**Description** You use the **ToFront** method of the Object object to move the object to the front.

**ABC Equivalent** The **ToFront** method is equivalent to clicking the Selection button, and clicking the To Front button.

## Related Topics

### Language Elements

[ToBack Method \(Chart Object\)](#)

[ToBack Method \(Object Object\)](#)

[ToFront Method \(Chart Object\)](#)

### Description

[Changing the Display Order of Objects](#)

### Object

[Object Object](#)

## OLE Property

Related

Related

Related

<b>Usage</b>	<i>Object</i> <i>Object</i> . <b>OLE</b>
<b>Description</b>	The <b>OLE</b> property lets you find or set the properties and methods associated with OLE objects. The <b>OLE</b> property is read only, but the properties from the object it returns are read/write.
<b>Data Type</b>	Object
<b>Value</b>	An OLE object
<b>ABC Equivalent</b>	None

## **Related Topics**

### **Language Elements**

[DoVerb Method](#)

[ObjectType Property](#)

### **Description**

[Using OLE Client Objects](#)

### **Object**

[Object Object](#)

## Related

# OLE Property and DoVerb Method Example

This example uses the **OLE** property of the Object object and the **DoVerb** method of the OLE object to execute an OLE verb.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object
Dim PaintHandle
Dim Pasted

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.New ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart
ABC.Visible = True                                     ' Make ABC visible

Clipboard.Clear                                        ' Empty the Clipboard
PaintHandle = Shell("PBRUSH.EXE", 1)                   ' Run Paintbrush
SendKeys "%FO", 1                                     ' Send File/Open to Paintbrush
SendKeys "c:\abc\samples\mgxlogo.bmp{ENTER}", 1
SendKeys "%EF", 1                                     ' Edit/Paste From
SendKeys "c:\abc\samples\mgxlogo.bmp{ENTER}", 1
SendKeys "%EC", 1                                     ' Copy to the Clipboard
Pasted = Chart.Paste                                  ' Paste the OLE object into ABC
If Not Pasted Then                                    ' If nothing was pasted, post an error
message and stop the code
    MsgBox "Either Paintbrush or the MGXLOGO.BMP file was not found. Please edit the code and try
again."
    Exit Sub
End If

ABC.Visible = True                                    ' Bring ABC to the front
Chart.View = 1                                        ' View the current page in ABC
Set Everything = Chart.Objects                        ' Get all items in the chart
Do
    Set Current = Everything.ItemFromAll              ' Choose the next item
    If Current.Type = 4 Then                          ' If an OLE object is found
        MsgBox "An OLE object exists on the chart. Click OK to execute the OLE verb."
        Current.OLE.DoVerb                          ' execute its default verb
        Exit Sub                                     ' and stop the code
    End If
Loop While Current.Valid
```

## TextBlock Property

Related

Related

Related

**Usage** *Object*.**TextBlock**

**Description** The **TextBlock** property lets you find the properties of a block of text. The **TextBlock** property is read only, but all the properties from the object it returns are read/write.

**Data Type** Object

**Value** The properties of a block of text

**ABC Equivalent** None

## Related Topics

### Language Elements

[DrawTextBlock Method](#)

### Description

[Creating Text Blocks](#)

### Object

[Object Object](#)

## Related

### TextBlock Property and AttachedToLine Property Example

This example uses the **TextBlock** property of the Object object and the **AttachedToLine** property of the TextBlock object to find a line that has text attached to it and turn the line red.

```
Dim ABC As Object, Chart As Object
Dim NewLine1 As Object, NewLine2 As Object
Dim NewText1 As Object, NewText2 As Object
Dim LineWithText As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set NewLine1 = Chart.DrawFreeLine(1, 4)
Chart.DrawPositionX = 2
Set NewLine2 = Chart.DrawFreeLine(2, 4)
Set NewText1 = Chart.DrawTextBlock("Attached text")
Set NewText2 = Chart.DrawTextBlock("Unattached text")
NewLine2.Line_.AttachText NewText1
Chart.View = 2

Set LineWithText = NewText1.TextBlock.AttachedToLine
LineWithText.Line_.StemColor = ABC.RED
MsgBox "The red line has text attached to it."
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Draw a line  
' Set a horizontal drawing position  
' Draw a line  
' Draw text objects  
  
' Attach a text object to a line  
' View used pages  
  
' Get the line with text attached  
' Make the line red

## TextLF Property

Related

Related

Related

### Usage

*ObjectObject.TextLF = TextString*

### Description

You use the **TextLF** property of the Object object to add or read text inside any shape or text block. When adding text, the property is identical to the **Text** property. When reading text, the property does not substitute spaces for Returns as the **Text** property does. If you do not wish to preserve Returns, you should use the **Text** property. The **TextLF** property is read/write.

### Data Type

String

### Value

The text inside a shape with the Returns preserved

### ABC Equivalent

None

## Related Topics

### Language Elements

[FitShapeToText Method](#)

[Paste Method](#)

[Text Property](#)

[Text Property \(Menu Collection\)](#)

[Text Property \(MenuItem Object\)](#)

### Description

[Adding Text to Shapes](#)

[Adding Text to a Shape](#)

### Object

[Object Object](#)



## Objects Collection

**Description** The Objects collection is below the Chart object. Below the Objects collection are the Object objects. You can have multiple Object objects in the collection.

### Properties

---

[Application](#)

[Count](#)

[Parent](#)

### Methods

---

[Item](#)

[ItemFromAll](#)

[ItemFromAttachments](#)

[ItemFromLines](#)

[ItemFromFieldValue](#)

[ItemFromNumber](#)

[ItemFromShapes](#)

[ItemFromSelection](#)

[ItemFromText](#)

[ItemFromUniqueID](#)

[ResetSearch](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Count Property

Related

Related

Related

### Usage

*Collection.Count*

### Description

The **Count** property returns the number of items in a collection. The collections in ABC OLE Automation are the Objects collection, Charts collection, FieldTemplates collection, FieldValues collection, and Menu collection. You often use the **Count** property in a loop along with the **Item** method and one of the ItemFrom methods to search through a collection. The **Count** property is read only.

### Data Type

Long

### Value

The number of items in a collection

### ABC Equivalent

None

## Related Topics

### Language Elements

[Item Method \(Charts Collection\)](#)  
[Item Method \(FieldTemplates Collection\)](#)  
[Item Method \(FieldValues Collection\)](#)  
[Item Method \(Menu Collection\)](#)  
[Item Method \(Objects Collection\)](#)  
[ItemFromAll Method](#)  
[ItemFromAttachments Method](#)  
[ItemFromFieldValue Method](#)  
[ItemFromLines Method](#)  
[ItemFromNumber Method](#)  
[ItemFromSelection Method](#)  
[ItemFromShapes Method](#)  
[ItemFromText Method](#)  
[ItemFromUniqueID Method](#)  
[ResetSearch Method](#)  
[Valid Property](#)

### Description

[Activating a Chart](#)  
[Finding the Total Number of Objects](#)

### Object

[Charts Collection](#)  
[FieldTemplates Collection](#)  
[FieldValues Collection](#)  
[Menu Collection](#)  
[Objects Collection](#)

**Related**

## Count Property Example

This example uses the **Count** property of the Objects collection to find how many objects are in the chart. Also see the example included with the [Add method of the FieldTemplates collection](#) and the second example included with the [Open method of the Charts collection](#).

```
Dim ABC As Object, Chart As Object
Dim Everything As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Everything = Chart.Objects                          ' Get all objects in the chart
MsgBox "The current chart contains " + Everything.Count + " objects."
```

## Item Method (Objects Collection)

Related

Related

Related

<b>Usage</b>	<i>ObjectsCollection.Item (Count)</i> The <i>Count</i> element is the index of the item within the collection.
<b>Description</b>	Use the <b>Item</b> method of the Objects collection to access Object objects within the Objects collection.
<b>Data Type</b>	Object
<b>Value</b>	The next valid Object object in the collection. If that object does not exist, the method returns Null.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Item Method \(Charts Collection\)](#)

[Item Method \(FieldTemplates Collection\)](#)

[Item Method \(FieldValues Collection\)](#)

[Item Method \(Menu Collection\)](#)

### Description

[Finding the Total Number of Objects](#)

### Object

[Objects Collection](#)



## ItemFromAll Method

Related

Related

Related

### Usage

*ObjectsCollection.ItemFromAll*

### Description

You use the **ItemFromAll** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

### Value

The **ItemFromAll** method returns successive objects from the Objects collection.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAttachments Method](#)

[ItemFromFieldValue Method](#)

[ItemFromLines Method](#)

[ItemFromNumber Method](#)

[ItemFromSelection Method](#)

[ItemFromShapes Method](#)

[ItemFromText Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

**Related**

## ItemFromAll Method and Type Property (Object Object) Example

This example uses the **ItemFromAll** method of the Objects collection and the **Type** property of the Object object to identify the types of the items in a chart.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object
Dim OriginalColor As Long

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Everything = Chart.Objects
Do
    Set Current = Everything.ItemFromAll
    OriginalColor = Current.Color
    Current.Color = ABC.MakeRGB(255, 64, 0)
    Select Case Current.Type
        Case 0
            MsgBox "The orange object is a shape."
        Case 1
            MsgBox "The orange object is a line."
        Case 2
            MsgBox "The orange object is text."
        Case 3
            MsgBox "The current object is a bitmap."
        Case 4
            MsgBox "The current object is an OLE client object."
        Case 5
            MsgBox "The orange object is a master item."
    End Select
    Current.Color = OriginalColor
Loop While Current.Valid
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' Get all objects in the chart

' Check each item
' Remember the object's original color
' Make the current object orange
' Determine the item's type

' Restore the original color
```

## ItemFromAttachments Method

Related

Related

Related

### Usage

*ObjectsCollection*.**ItemFromAttachments** (*ObjectWithAttachment1* [, *ObjectWithAttachment2*])

The *ObjectWithAttachment1* element is any object that you want to include in the search.

The *ObjectWithAttachment2* element, which is optional, is any object that you want to include in the search.

### Description

You use the **ItemFromAttachments** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

### Value

The **ItemFromAttachments** method returns (from the Objects collection) successive shape, text, or line objects that are attached to the one or two objects you specify.

### ABC Equivalent

None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAll Method](#)

[ItemFromFieldValue Method](#)

[ItemFromLines Method](#)

[ItemFromNumber Method](#)

[ItemFromSelection Method](#)

[ItemFromShapes Method](#)

[ItemFromText Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

## Related

### ItemFromAttachments Method Example

This example uses the **ItemFromAttachments** method of the Objects collection to find a line that has text attached to it.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Line1 As Object, Text1 As Object
Dim Count As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")
Set Chart = ABC.ActiveChart
ABC.Visible = True

Set Shape1 = Chart.DrawShape("Connector")
Set Shape2 = Chart.DrawShape("Connector")
Set Line1 = Chart.DrawLine(Shape1, Shape2)
Line1.Repaint
Set Text1 = Chart.DrawTextBlock("Going my way?")
Text1.Font.Opaque = True
Line1.Line_.AttachText Text1

Set Everything = Chart.Objects
Set Current = Everything.ItemFromAttachments(Text1)
Current.Color = ABC.Red
ABC.MsgBox ("The red object has text attached.")
```

```
' Start ABC
' Get the active chart
' Make ABC visible

' Draw shapes

' Draw a line connecting the shapes
' Refresh the screen
' Create a text object
' Make the text background opaque
' Attach the text to the line

' Get all objects in the chart
' Find item with text attached
' Make the item red
```

## ItemFromLines Method

Related

Related

Related

### Usage

*ObjectsCollection*.ItemFromLines

### Description

You use the **ItemFromLines** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

### Value

The **ItemFromLines** method returns, from the Objects collection, successive lines.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAll Method](#)

[ItemFromAttachments Method](#)

[ItemFromFieldValue Method](#)

[ItemFromNumber Method](#)

[ItemFromSelection Method](#)

[ItemFromShapes Method](#)

[ItemFromText Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

## Related

# ItemFromLines Method, Source Property, and Destination Property Example

This example uses the **ItemFromLines** method of the Objects collection and the **Source** property and **Destination** property of the Line\_ object to find a line, its source shape, and its destination shape. This example assumes that two shapes connected by a line already exist on the current chart.

```
Dim ABC As Object, Chart As Object
Dim Line1 As Object
Dim StartShape As Object, EndShape As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Chart.Select (1)
Set Line1 = Chart.Objects.ItemFromLines
Set StartShape = Line1.Line_.Source
Set EndShape = Line1.Line_.Destination

StartShape.Text = "Source"
EndShape.Text = "Destination"
```

```
' Start ABC
' Make ABC visible
' Get the active chart

' Select the lines in the chart
' Assign the line to the variable Line1
' Find the line's source shape
' Find the line's destination shape

' "Source" in source shape
' "Destination" in destination shape
```

## ItemFromFieldValue Method

Related

Related

Related

<b>Usage</b>	<i>ObjectsCollection.ItemFromFieldValue (FieldTemplateObject, Value)</i> The <i>FieldTemplateObject</i> element is any FieldTemplate object that you want to examine. The <i>Value</i> element is the value of the FieldTemplate object that you are searching for. The <i>Value</i> element is a double or a string, as appropriate for the FieldTemplate object.
<b>Description</b>	You use the <b>ItemFromFieldValue</b> method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the <b>Valid</b> property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.
<b>Value</b>	The <b>ItemFromFieldValue</b> method returns, from the Objects collection, successive objects that contain a field with the value you specify.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAll Method](#)

[ItemFromAttachments Method](#)

[ItemFromLines Method](#)

[ItemFromNumber Method](#)

[ItemFromSelection Method](#)

[ItemFromShapes Method](#)

[ItemFromText Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

[Knowing When Data Fields Change](#)

### Object

[Objects Collection](#)

**Related**

## ItemFromFieldValue Method and MsgBox Method Example

This example uses the **ItemFromFieldValue** method of the Objects collection to find the correct answer to a guessing game. It uses the **MsgBox** method of the Application object to give the answer.

```
Dim ABC As Object, Chart As Object
Dim Field1 As Object, NewShape As Object
Dim Count As Single, Answer As Single
Dim NumberIn As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
ABC.New ' Add a new chart
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Field1 = Chart.FieldTemplates.Add("Magic Number") ' Create a field
Field1.Type = 5                                       ' Make the field's type number

For Count = 1 To 3
    Set NewShape = Chart.DrawShape("Operation")        ' Draw a shape
    NumberIn = Int(InputBox("Enter a number between 1 and 10.")) ' Accept user input
    NewShape.FieldValues.Item("Magic Number").Value = NumberIn ' Assign to field
Next Count

Randomize
Answer = Int(10 * Rnd + 1)                             ' Randomly generate an integer
Set Everything = Chart.Objects                         ' Get all objects in the chart
Do
    ' Find field value equal to Answer
    Set Current = Everything.ItemFromFieldValue(Field1; "Answer")
    Current.Text = "You win!"                          ' Enter text into the shape
    Current.Color = ABC.Red                            ' Make the shape red
Loop While Current.Valid
ABC.MsgBox ("Thanks for playing! The correct answer was " + Answer + ".")
```

## ItemFromNumber Method

Related

Related

Related

<b>Usage</b>	<i>ObjectsCollection.ItemFromNumber (Value)</i> The <i>Value</i> element is a string or double that is the number or identifier of the shape you are searching for.
<b>Description</b>	You use the <b>ItemFromNumber</b> method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the <b>Valid</b> property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.
<b>Value</b>	The <b>ItemFromNumber</b> method returns, from the Objects collection, successive shapes with the number you specify.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAll Method](#)

[ItemFromAttachments Method](#)

[ItemFromFieldValue Method](#)

[ItemFromLines Method](#)

[ItemFromSelection Method](#)

[ItemFromShapes Method](#)

[ItemFromText Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

## Related

### ItemFromNumber Method Example

This example uses the **ItemFromNumber** method of the Objects collection to find a specific shape by its number.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Count As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")
Set Chart = ABC.ActiveChart
ABC.Visible = True

Chart.NextNumber = 100
For Count = 1 To 4
    Set NewShape = Chart.DrawShape("Operation")
Next Count

Set Everything = Chart.Objects
Set Current = Everything.ItemFromNumber(102)
Current.Shape.FillPattern = 22
```

```
' Start ABC
' Get the active chart
' Make ABC visible

' Set the next shape number used
' Draw four operation shapes

' Get all objects in the chart
' Get shape 102
' Fill the shape with a pattern
```

## ItemFromShapes Method

Related

Related

Related

**Usage** *ObjectsCollection.ItemFromShapes*

**Description** You use the **ItemFromShapes** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

**Value** The **ItemFromShapes** method returns successive shapes from the Objects collection.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAll Method](#)

[ItemFromAttachments Method](#)

[ItemFromFieldValue Method](#)

[ItemFromLines Method](#)

[ItemFromNumber Method](#)

[ItemFromSelection Method](#)

[ItemFromText Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

[Selecting Shapes](#)

### Object

[Objects Collection](#)

## Related

# ItemFromShapes Method, IsLinked Property, Number Property, and LinkedChartName Property Example

This example uses the **ItemFromShapes** method of the Objects collection, the **IsLinked** property, **Number** property, and **LinkedChartName** property of the Shape object to select shapes, find the shapes that are linked, and describe them in a message box. For the program to work usefully, the chart must contain shapes, with at least one of the shapes linked.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim CurrentShape As Object
Dim SelectedShapes As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set SelectedShapes = Chart.Objects

Do
    Set CurrentShape = SelectedShapes.ItemFromShapes    ' Check all shapes in the chart
    If CurrentShape.Shape.IsLinked Then                 ' If shape is linked, display a message
        MsgBox "Shape #" + CurrentShape.Shape.Number + " is linked to " +
            CurrentShape.Shape.LinkedChartName
    End If
Loop While CurrentShape.Valid
```

## ItemFromSelection Method

Related

Related

Related

Related

### Usage

*ObjectsCollection*.**ItemFromSelection**

### Description

You use the **ItemFromSelection** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

### Value

The **ItemFromSelection** method returns, from the Objects collection, successive selected objects.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAll Method](#)

[ItemFromAttachments Method](#)

[ItemFromFieldValue Method](#)

[ItemFromLines Method](#)

[ItemFromNumber Method](#)

[ItemFromShapes Method](#)

[ItemFromText Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

## Related

# ItemFromSelection Method, Top Property, Left Property, Right Property, and Bottom Property Example 1

This example uses the **ItemFromSelection** method of the Objects collection and the **Top** property of the Object object to find objects and put all their upper edges in the same place. If you wish, you can substitute the **Left** property, **Right** property, or **Bottom** property for the **Top** property to achieve a similar effect. For this program to work, the chart must contain two or more shapes.

```
Dim ABC As Object, Chart As Object
Dim SelectedItems As Object
Dim CurrentItem As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set SelectedItems = Chart.Objects

Chart.Select (0)

Do
    Set CurrentItem = SelectedItems.ItemFromSelection
    CurrentItem.Top = 1
Loop While CurrentItem.Valid
```

' Everything in the chart  
' Currently selected shape  
  
' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Use all shapes in the current chart  
  
' Select all shapes in the chart  
  
' Place top edge of items at 1 inch

**Related**

## ItemFromSelection Method, Valid Method, CenterX Property, and CenterY Property Example 2

This example uses the **ItemFromSelection** method of the Objects collection and the **Valid** property, **CenterX** property, and **CenterY** property of the Object object to find selected objects, ensure that only valid objects are acted on, and find and report the center of the objects both horizontally and vertically.

```
Dim ABC As Object, Chart As Object
Dim CurrentItem As Object
Dim Center_X, Center_Y As String
Dim SelectedItems As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set SelectedItems = Chart.Objects                       ' Get all objects in the chart

Set CurrentItem = SelectedItems.ItemFromSelection      ' Get the first selected object
Do While CurrentItem.Valid                             ' Loop through all selected objects
    Center_X = CStr(CurrentItem.CenterX)              ' Horizontal center of object
    Center_Y = CStr(CurrentItem.CenterY)              ' Vertical center of object
    MsgBox "X Value = " + Center_X                    ' Display the X coordinate
    MsgBox "Y Value = " + Center_Y                    ' Display the Y coordinate
    Set CurrentItem = SelectedItems.ItemFromSelection ' Loop through all objects
Loop
```

## ItemFromText Method

Related

Related

Related

### Usage

*ObjectsCollection*.**ItemFromText** (*Text*)

The *Text* element is the text string you are searching for.

### Description

You use the **ItemFromText** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

### Value

The **ItemFromText** method returns, from the Objects collection, successive objects that contain the text you specify.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Item Method \(Objects Collection\)](#)

[ItemFromAll Method](#)

[ItemFromAttachments Method](#)

[ItemFromFieldValue Method](#)

[ItemFromLines Method](#)

[ItemFromNumber Method](#)

[ItemFromSelection Method](#)

[ItemFromShapes Method](#)

[ItemFromUniqueID Method](#)

[ResetSearch Method](#)

[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

## Related

### ItemFromText Method Example

This example uses the **ItemFromText** method of the Objects collection to find a specific shape by its text.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, Shape2 As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")
Set Chart = ABC.ActiveChart
ABC.Visible =

Set Shape1 = Chart.DrawShape
Shape1.Text = "Roses are red"
Set Shape2 = Chart.DrawShape
Shape2.Text = "Violets are blue"
Shape2.Font.Color = ABC.White

Set Everything = Chart.Objects
Set Current = Everything.ItemFromText("Violets")
Current.Shape.FillColor = ABC.MakeRGB(64, 0, 127)
```

```
' Start ABC
' Get the active chart
' Make ABC visible

' Draw a shape
' Enter text in the shape
' Draw a shape
' Enter text in the shape
' Set the text color

' Get all objects in the chart
' Get the shape containing "Violets"
' Fill the shape with purple
```

## ItemFromUniqueID Method

Related

Related

Related

### Usage

*ObjectsCollection*.**ItemFromUniqueID** (*UniqueID*)

The *UniqueID* element is the unique identification number of the object you are searching for.

### Description

You use the **ItemFromUniqueID** method and the other ItemFrom methods to find objects in a chart. You use them in a loop, most often a Do While loop. In the While part of the loop, you use the **Valid** property to check only valid objects. Each time the loop executes, the method returns the next object so you can test the objects for a property value and act on the objects that meet that value.

### Value

The **ItemFromUniqueID** method returns, from the Objects collection, the object with the unique identification number you specify.

**ABC Equivalent** None

## Related Topics

### Language Elements

[ItemFromAll Method](#)  
[ItemFromShapes Method](#)  
[ItemFromLines Method](#)  
[ItemFromSelection Method](#)  
[ItemFromFieldValue Method](#)  
[ItemFromAttachments Method](#)  
[ItemFromNumber Method](#)  
[Item Method \(Objects Collection\)](#)  
[ItemFromText Method](#)  
[ResetSearch Method](#)  
[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

## Related

# ItemFromUniqueID Method Example

This example uses the **ItemFromUniqueID** method of the Objects collection to identify a specific object.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object, FirstYellow As Object
Dim Shape3 As Object, SecondYellow As Object
Dim Everything As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
ABC.New ' Add a new chart
Set Chart = ABC.ActiveChart

Chart.DrawSpacingX = .5
Set Shape1 = Chart.DrawShape("Decision")
Shape1.Color = ABC.Red
Set FirstYellow = Chart.DrawShape("Decision")
FirstYellow.Color = ABC.Yellow
Set Shape3 = Chart.DrawShape("Decision")
Shape3.Color = ABC.Blue

Chart.Select (0)
Chart.Duplicate
Chart.Clear_

Set Everything = Chart.Objects
' Get the yellow shape from the duplicate set
Set SecondYellow = Everything.ItemFromUniqueID(FirstYellow.UniqueID + 3)
FirstYellow.ToBack
SecondYellow.ToFront
SecondYellow.Text = "UniqueID #" + CStr(SecondYellow.UniqueID) ' Display the ID
SecondYellow.Shape.FitShapeToText ' Enlarge the shape so the text fits
```

## ResetSearch Method

Related

Related

Related

**Usage** *ObjectsCollection.ResetSearch*

**Description** You use the **ResetSearch** method to reset all searches that use the ItemFrom methods to the beginning of the items in the chart.

**ABC Equivalent** None

## Related Topics

### Language Elements

[ItemFromAll Method](#)  
[ItemFromAttachments Method](#)  
[ItemFromFieldValue Method](#)  
[ItemFromLines Method](#)  
[ItemFromNumber Method](#)  
[ItemFromSelection Method](#)  
[ItemFromShapes Method](#)  
[ItemFromText Method](#)  
[ItemFromUniqueID Method](#)  
[Valid Property](#)

### Description

[Finding Objects in a Chart](#)

### Object

[Objects Collection](#)

## Related

# ResetSearch Method Example

This example uses the **ResetSearch** method of the Objects collection to start a subsequent search through shapes from the beginning.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Count As Long
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.New ' Add a new chart
Set Chart = ABC.ActiveChart                             ' Get the active chart
ABC.Visible = True                                     ' Make ABC visible
Chart.View = 2                                         ' Set the view to Used Pages

Chart.NextNumber = 100                                 ' Set the next shape number used
For Count = 1 To 10
    Set NewShape = Chart.DrawShape                       ' Draw a shape
    NewShape.Shape.FillPattern = Count                   ' Fill the shape with a pattern
Next Count

Set Everything = Chart.Objects                          ' Get all objects in the chart
Do
    Set Current = Everything.ItemFromShapes              ' Get the next shape
    Current.Shape.FillColor = ABC.Red                   ' Make the shape's fill red
Loop Until Current.Shape.FillPattern = 5                ' Search until you find pattern 5
MsgBox ("The first search reached fill pattern 5.")

Everything.ResetSearch                                 ' Start next search from beginning
Do
    Set Current = Everything.ItemFromShapes              ' Get the next shape
    Current.Shape.BorderColor = ABC.Blue                ' Make the shape's border blue
Loop Until Current.Shape.Number = "108"                ' Search until you find shape 108
MsgBox ("The second search started over in order to find shape 108.")
```

## OLE Object

**Description** The OLE object is below the Object object. Note that this is an OLE client object from another application, not an ABC OLE Automation object. You can have only one OLE object for each Object object.

### Properties

---

[Application](#)  
[ObjectType](#)  
[Parent](#)

### Methods

---

[DoVerb](#)  
[RestorePicture](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## ObjectType Property

Related

Related

Related

<b>Usage</b>	<i>OLEObject.ObjectType</i>
<b>Description</b>	The <b>ObjectType</b> property lets you find the short object class name of an object that is embedded or linked. The <b>ObjectType</b> property is read only.
<b>Data Type</b>	String
<b>Value</b>	The short object class name of an object that is embedded or linked. The name depends on the OLE server. For example, the name for bitmaps might be "Paintbrush Picture" and the name for Word for Windows is "Document."
<b>ABC Equivalent</b>	None

## **Related Topics**

### **Language Elements**

[DoVerb Method](#)

[InsertObjectFromFile Method](#)

[OLE Property](#)

[PasteLink Method](#)

[UpdateFields Method](#)

### **Description**

[Using OLE Client Objects](#)

### **Object**

[OLE Object](#)

## Related

### ObjectType, Objects Properties Example

This example uses the **ObjectType** property of the OLE object and the **Objects** property of the Chart object to find the type of an OLE object.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set Everything = Chart.Objects                     ' Get all items in the chart
Do
    Set Current = Everything.ItemFromAll           ' Choose the next item
    ' Display the OLE type
    MsgBox "Current object's OLE type is "" + Current.OLE.ObjectType + """"
Loop While Current.Valid And Current.Type <> 5    ' Skip over master items
```

## DoVerb Method

Related

Related

Related

**Usage** *OLEObject.DoVerb* ([Verb])

**Description** You use the **DoVerb** method to specify an OLE verb to execute if the object is a linked or embedded OLE object. If you do not specify a verb, the default verb is used.

**ABC Equivalent** None.

## Related Topics

### Language Elements

[InsertObjectFromFile Method](#)

[ObjectType Property](#)

[OLE Property](#)

[PasteLink Method](#)

[UpdateFields Method](#)

### Description

[Using OLE Client Objects](#)

### Object

[OLE Object](#)

## RestorePicture Method (OLE Object)

Related

Related

Related

### Usage

*OLEObject*.RestorePicture

### Description

The **RestorePicture** method of the OLE object lets you restore OLE client objects to their original size. This method is nearly the same as the Object object's **RestorePicture** method. The difference is that this method only works on OLE objects, while the Object object's **RestorePicture** method handles bitmaps as well as OLE objects.

**ABC Equivalent** None

## Related Topics

### Language Elements

[RestorePicture Method \(Object Object\)](#)

### Description

[Restoring OLE Objects](#)

### Object

[OLE Object](#)

## Related

### RestorePicture Method (OLE Object) Example

This example uses the **RestorePicture** method of the OLE object to restore all OLE objects to their original size. It assumes you already have an OLE client object on the current chart.

```
Dim ABC As Object, Chart As Object
Dim Everything As Object, Current As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Everything = Chart.Objects
Do
    Set Current = Everything.ItemFromAll
    Current.OLE.RestorePicture
Loop While Current.Valid

' Start ABC
' Make ABC visible
' Get the active chart

' Get all items in the chart

' Choose the next item
' Return OLE objects to original size
```

## PageLayout Object

**Description** The PageLayout object is below the Chart object. You can have only one PageLayout object.

### **Properties**

[Application](#)  
[Height](#)  
[MarginBottom](#)  
[MarginLeft](#)  
[MarginRight](#)  
[MarginTop](#)  
[Orientation](#)  
[PageHeight](#)  
[PageOrder](#)  
[PageWidth](#)  
[PaperSize](#)  
[Parent](#)  
[PrintBlankPages](#)  
[Width](#)

### **Methods**

There are no methods for the PageLayout object.

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## Height Property (PageLayout Object)

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject.Height</i>
<b>Description</b>	You use the <b>Height</b> property of the PageLayout object to find the height of the drawing area. The <b>Height</b> property is read only.
<b>Data Type</b>	Double
<b>Value</b>	The height of the drawing area in pixels
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[Height Property \(Application Object\)](#)

[Height Property \(Object Object\)](#)

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PageOrder Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## MarginBottom Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject.MarginBottom = Distance</i>
<b>Description</b>	You use the <b>MarginBottom</b> property to find or set the bottom page margins. The <b>MarginBottom</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The bottom page margin in the current units
<b>ABC Equivalent</b>	The <b>MarginBottom</b> property is equivalent to choosing PageLayout in the File menu and entering a number in the Margin Bottom text box.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## MarginLeft Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject.MarginLeft = Distance</i>
<b>Description</b>	You use the <b>MarginLeft</b> property to find or set the left page margins. The <b>MarginLeft</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The left page margin in the current units
<b>ABC Equivalent</b>	The <b>MarginLeft</b> property is equivalent to choosing PageLayout in the File menu and entering a number in the Margin Left text box.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginBottom Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## MarginRight Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject.MarginRight = Distance</i>
<b>Description</b>	You use the <b>MarginRight</b> property to find or set the right page margin. The <b>MarginRight</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The right page margin in the current units
<b>ABC Equivalent</b>	The <b>MarginRight</b> property is equivalent to choosing PageLayout in the File menu and entering a number in the Margin Left text box.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## MarginTop Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject.MarginTop = Distance</i>
<b>Description</b>	You use the <b>MarginTop</b> property to find or set the top page margin. The <b>MarginTop</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The top page margin in the current units
<b>ABC Equivalent</b>	The <b>MarginTop</b> property is equivalent to choosing PageLayout in the File menu and entering a number in the Margin Top text box.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## Orientation Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject.Orientation = Value</i>
<b>Description</b>	You use the <b>Orientation</b> property to find or set portrait or landscape orientation for the page. The <b>Orientation</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The value in the <b>Orientation</b> property indicates the page orientation. <b>Value Orientation</b> 0 Portrait 1 Landscape
<b>ABC Equivalent</b>	The <b>Orientation</b> property is equivalent to choosing the Page Layout command in the File menu and selecting the page layout.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[PageHeight Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## Related

# PageLayout Properties Example

This example uses properties of the PageLayout object and the **PageLayout** property of the Chart object to set up the ABC page.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
ABC.New
Set Chart = ABC.ActiveChart

Chart.Units = 0

Chart.PageLayout.Orientation = 1

Chart.PageLayout.MarginLeft = 0
Chart.PageLayout.MarginRight = 0
Chart.PageLayout.MarginTop = 0
Chart.PageLayout.MarginBottom = 0

If Chart.PageLayout.Width > 7 Then
    Chart.PageLayout.PageWidth = 7
End If

If Chart.PageLayout.Height > 5 Then
    Chart.PageLayout.PageHeight = 5
End If
```

```
' Start ABC
' Make ABC visible
' Add a new chart
' Get the active chart

' Set units to inches

' Set landscape page orientation

' Set left margin
' Set right margin
' Set top margin
' Set bottom margin

' Check current page width
' Make pages 7" wide

' Check current page height
' Make pages 5" high
```

## PageHeight Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject</i> . <b>PageHeight</b> = <i>Distance</i>
<b>Description</b>	You use the <b>PageHeight</b> property to find or set the height of the page. The <b>PageHeight</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The height of the page in the current units
<b>ABC Equivalent</b>	The <b>PageHeight</b> property is equivalent to choosing Page Layout in the File menu and entering a number in the Paper Size Height text box.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## PageOrder Property

**Related**

**Related**

**Related**

### Usage

*PageLayoutObject*.**PageOrder** = *Order*

### Description

You use the **PageOrder** property to find or set the order in which to print the pages in the chart. The **PageOrder** property is read/write.

### Data Type

Integer

### Value

The value of the **PageOrder** property method determines the order to print the pages in the chart as shown in the following table.

Value	Description	Pattern
-------	-------------	---------

0	Across, then down	<b>Related</b>
---	-------------------	----------------

1 Down, then across

**Related**

**ABC Equivalent** The **PageOrder** property is equivalent to choosing Page Layout in the File menu and choosing one of the Page Order options.

## Related Topics

### Language Elements

[PrintBlankPages Property](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## PageWidth Property

Related

Related

Related

**Usage** *PageLayoutObject.PageWidth = Distance*

**Description** You use the **PageWidth** property to find or set the width of the page. The **PageWidth** property is read/write.

**Data Type** Double

**Value** The width of the page in the current units

**ABC Equivalent** The **PageWidth** property is equivalent to choosing Page Layout in the File menu and entering a value in the Paper Size Width text box.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PaperSize Property](#)

[PrintBlankPages Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

[Width Property \(Application Object\)](#)

[Width Property \(Object Object\)](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## PaperSize Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject.PaperSize = Size</i>
<b>Description</b>	You use the <b>PaperSize</b> property to find or set the size of paper to be printed. The program uses a "loose matching" routine when you are setting the value so, for example, setting the <b>PaperSize</b> property to "let" chooses the size "Letter 8 1/2 x 11 in." The <b>PaperSize</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The size of the paper
<b>ABC Equivalent</b>	The <b>PaperSize</b> property is equivalent to choosing Page Layout in the File menu and selecting a value in the Paper Size list box.

## Related Topics

### Language Elements

[Height Property \(PageLayout Object\)](#)

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PageWidth Property](#)

[Width Property \(PageLayout Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## Related

# PaperSize, PageOrder, PrintBlankPages Properties Example

This example uses the **PaperSize** property, the **PageOrder** property, and the **PrintBlankPages** property of the **PageLayout** object and the **PageLayout** property of the **Chart** object to prepare a chart for printing.

```
Dim ABC As Object, Chart As Object
Dim NewShape As Object
Dim Printed

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
ABC.New
Set Chart = ABC.ActiveChart

Chart.PageLayout.PaperSize = "Letter"
Chart.PageLayout.PageOrder = 1
Chart.PageLayout.PrintBlankPages = False

Chart.DrawPositionX = 3.5
Chart.DrawPositionY = 4.75
Set NewShape = Chart.DrawShape
NewShape.Text = "Page one"

Chart.DrawPositionX = 10.5
Set NewShape = Chart.DrawShape
NewShape.Text = "Page three. Page two is blank."
NewShape.Shape.FitShapeToText

Chart.DrawPositionX = 10.5
Chart.DrawPositionY = 14.25
Set NewShape = Chart.DrawShape
NewShape.Text = "Page four"

Chart.View = 2

ABC.Printer = "LPT1"
Printed = Chart.PrintOut
```

```
' Start ABC
' Make ABC visible
' Add a new chart
' Get the active chart

' Use a Letter 8.5 x 11 size page
' Print pages down then across
' Omit printing pages with no objects

' Set X location for the first shape
' Set Y location for the first shape
' Place the first shape on page 1
' Enter text in the shape

' Set X location for the next shape
' Place the second shape on page 3
' Enter text in the shape
' Enlarge the shape if necessary

' Set X location for the next shape
' Set Y location for the next shape
' Place the third shape on page 4
' Enter text in the shape

' Display the used pages

' Select the first printer on LPT1
' Print the chart
```

## PrintBlankPages Property

Related

Related

Related

<b>Usage</b>	<i>PageLayoutObject</i> . <b>PrintBlankPages</b> = {True   False}
<b>Description</b>	You use the <b>PrintBlankPages</b> property to specify whether a blank page should be printed if there are no objects on the page. The <b>PrintBlankPages</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True prints a blank page when there is a blank page in the chart; False prints only pages that have objects on them.
<b>ABC Equivalent</b>	The <b>PrintBlankPages</b> property is equivalent to choosing Page Layout in the File menu and selecting or deselecting the Print Blank Pages option.

## Related Topics

Language Elements

[PageOrder Property](#)

Description

[Printing Charts](#)

Object

[PageLayout Object](#)

## Width Property (PageLayout Object)

Related

Related

Related

**Usage** *PageLayoutObject.Width*

**Description** You use the **Width** property to find or set the width of the drawing area. The **Width** property is read only.

**Data Type** Double

**Value** The width of the drawing area

**ABC Equivalent** None

## Related Topics

### Language Elements

[MarginBottom Property](#)

[MarginLeft Property](#)

[MarginRight Property](#)

[MarginTop Property](#)

[Orientation Property](#)

[PageHeight Property](#)

[PageWidth Property](#)

[PaperSize Property](#)

[Width Property \(Application Object\)](#)

[Width Property \(Object Object\)](#)

### Description

[Adjusting the Page Layout](#)

### Object

[PageLayout Object](#)

## Preferences Object

**Description** The Preferences object is below the Application object. You can have only one Preferences object.

### **Properties**

[AlignToRulers](#)

[Application](#)

[ChannelAlignment](#)

[LineSpacingX](#)

[LineSpacingY](#)

[Parent](#)

[ShapeSizing](#)

[ShowRulers](#)

[SmartShapeSpacing](#)

[SSSHorizontal](#)

[SSSVERTICAL](#)

[TouchAlignment](#)

[Units](#)

### **Methods**

There are no methods for the Preferences object.

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## AlignToRulers Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.AlignToRulers = Value</i>								
<b>Description</b>	The <b>AlignToRulers</b> property lets you find or set the choices for aligning to rulers. The <b>AlignToRulers</b> property is read/write.								
<b>Data Type</b>	Integer								
<b>Value</b>	The values for the <b>AlignToRulers</b> property are in the following table.								
	<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Off (not selected)</td></tr><tr><td>1</td><td>Coarse</td></tr><tr><td>2</td><td>Fine</td></tr></tbody></table>	Value	Description	0	Off (not selected)	1	Coarse	2	Fine
Value	Description								
0	Off (not selected)								
1	Coarse								
2	Fine								
<b>ABC Equivalent</b>	The <b>AlignToRulers</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and selecting or deselecting the Align to Rulers option, the Coarse option, or the Fine option.								

## Related Topics

### Language Elements

[ChannelAlignment Property](#)

[ShowRulers Property](#)

[SmartShapeSpacing Property](#)

[SSSHorizontal Property](#)

[SSSVertical Property](#)

[TouchAlignment Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Alignment Options](#)

### Object

[Preferences Object](#)

## Related

# Preferences Object Properties Example

This example uses properties of the Preferences object to set chart preferences.

```
Dim ABC As Object, Chart As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True

ABC.Preferences.AlignToRulers = 2
ABC.Preferences.ChannelAlignment = True
ABC.Preferences.LineSpacingX = 2
ABC.Preferences.LineSpacingY = 2
ABC.Preferences.ShapeSizing = 1
ABC.Preferences.ShowRulers = True
ABC.Preferences.SmartShapeSpacing = True
ABC.Preferences.SSSHORIZONTAL = 1
ABC.Preferences.SSSVERTICAL = 1
ABC.Preferences.TouchAlignment = True
ABC.Preferences.Units = 0

' Start ABC
' Make ABC visible

' Set fine ruler alignment
' Enable channel alignment
' Horizontal spacing for new lines
' Vertical spacing for new lines
' Last used
' Display the rulers
' Enable smart shape spacing
' Smart shape spacing horizontal value
' Smart shape spacing vertical value
' Enable touch alignment
' Set units to inches

MsgBox "Choose File and Preferences to verify the settings."
```

## ChannelAlignment Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.ChannelAlignment</i> = {True   False}
<b>Description</b>	The <b>ChannelAlignment</b> property lets you find or set whether channel alignment is turned on. The <b>ChannelAlignment</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True turns channel alignment on; False turns it off.
<b>ABC Equivalent</b>	The <b>ChannelAlignment</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and selecting the Channel Alignment option (True) or deselecting the option (False).

## Related Topics

### Language Elements

[AlignToRulers Property](#)  
[ShowRulers Property](#)  
[SmartShapeSpacing Property](#)  
[SSSHorizontal Property](#)  
[SSSVertical Property](#)  
[TouchAlignment Property](#)

### Description

[Alignment Options](#)

### Object

[Preferences Object](#)

## LineSpacingX Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.LineSpacingX = HorizontalLineSpacing</i>
<b>Description</b>	The <b>LineSpacingX</b> property lets you find or set the horizontal value used by line spacing. The <b>LineSpacingX</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The number of inches or centimeters that line spacing is to use horizontally
<b>ABC Equivalent</b>	The <b>LineSpacingX</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Line Spacing button, and entering a value for the horizontal component used by line spacing.

## Related Topics

### Language Elements

[LineSpacingY Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Line Options](#)

### Object

[Preferences Object](#)

## LineSpacingY Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.LineSpacingY = VerticalLineSpacing</i>
<b>Description</b>	The <b>LineSpacingY</b> property lets you find or set the vertical value used by line spacing. The <b>LineSpacingY</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The number of inches or centimeters that line spacing is to use vertically
<b>ABC Equivalent</b>	The <b>LineSpacingY</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Line Spacing button, and entering a value for the vertical component used by line spacing.

## Related Topics

### Language Elements

[LineSpacingX Property](#)

[Units Property \(Chart Object\)](#)

[Units Property \(Preferences Object\)](#)

### Description

[Line Options](#)

### Object

[Preferences Object](#)

## ShapeSizing Property

Related

Related

Related

**Usage** *PreferencesObject.ShapeSizing = Value*

**Description** The **ShapeSizing** property lets you find or set the choices for shape sizing. The **ShapeSizing** property is read/write.

**Data Type** Integer

**Value** The values for the **ShapeSizing** property are in the following table.

Value	Description
-------	-------------

0	Manual
---	--------

1	Last Used
---	-----------

2	Palette Size
---	--------------

**ABC Equivalent** The **ShapeSizing** property is equivalent to choosing Preferences in the ABC File menu, clicking the Shape Sizing button, and selecting the Manual option, the Last Used option, or the Palette Size option.

## Related Topics

### Description

[Shape Sizing Options](#)

### Object

[Preferences Object](#)

## ShowRulers Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject</i> . <b>ShowRulers</b> = {True   False}
<b>Description</b>	The <b>ShowRulers</b> property lets you find or set whether the rulers are shown. The <b>ShowRulers</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True turns the rulers on; False turns them off.
<b>ABC Equivalent</b>	The <b>ShowRulers</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and selecting the Show Rulers option (True) or deselecting the option (False).

## Related Topics

### Language Elements

[AlignToRulers Property](#)  
[ChannelAlignment Property](#)  
[SmartShapeSpacing Property](#)  
[SSSHorizontal Property](#)  
[SSSVertical Property](#)  
[TouchAlignment Property](#)

### Description

[Alignment Options](#)

### Object

[Preferences Object](#)

## SmartShapeSpacing Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject</i> . <b>SmartShapeSpacing</b> = {True   False}
<b>Description</b>	The <b>SmartShapeSpacing</b> property lets you find or set whether smart shape spacing is turned on. The <b>SmartShapeSpacing</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True turns smart space shaping on; False turns it off.
<b>ABC Equivalent</b>	The <b>SmartShapeSpacing</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and selecting the Smart Shape Spacing option (True) or deselecting the option (False).

## Related Topics

### Language Elements

[AlignToRulers Property](#)

[ChannelAlignment Property](#)

[ShowRulers Property](#)

[SSSHorizontal Property](#)

[SSSVertical Property](#)

[TouchAlignment Property](#)

### Description

[Alignment Options](#)

### Object

[Preferences Object](#)

## SSSHorizontal Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.SSSHORIZONTAL = HorizontalValue</i>
<b>Description</b>	The <b>SSSHORIZONTAL</b> property lets you find or set the horizontal value used by Smart Shape Spacing. The <b>SSSHORIZONTAL</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The number of inches or centimeters that Smart Shape Spacing is to use horizontally
<b>ABC Equivalent</b>	The <b>SSSHORIZONTAL</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and entering a value for the horizontal component used by Smart Shape Spacing.

## Related Topics

### Language Elements

- [AlignToRulers Property](#)
- [ChannelAlignment Property](#)
- [ShowRulers Property](#)
- [SmartShapeSpacing Property](#)
- [SSSVERTICAL Property](#)
- [TouchAlignment Property](#)
- [Units Property \(Chart Object\)](#)
- [Units Property \(Preferences Object\)](#)

### Description

- [Alignment Options](#)

### Object

- [Preferences Object](#)

## SSSVertical Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.SSSVertical = VerticalValue</i>
<b>Description</b>	The <b>SSSVertical</b> property lets you find or set the vertical value used by Smart Shape Spacing. The <b>SSSVertical</b> property is read/write.
<b>Data Type</b>	Double
<b>Value</b>	The number of inches or centimeters that Smart Shape Spacing is to use vertically
<b>ABC Equivalent</b>	The <b>SSSVertical</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and entering a value for the vertical component used by Smart Shape Spacing.

## Related Topics

### Language Elements

- [AlignToRulers Property](#)
- [ChannelAlignment Property](#)
- [ShowRulers Property](#)
- [SmartShapeSpacing Property](#)
- [SSSHorizontal Property](#)
- [TouchAlignment Property](#)
- [Units Property \(Chart Object\)](#)
- [Units Property \(Preferences Object\)](#)

### Description

- [Alignment Options](#)

### Object

- [Preferences Object](#)

## TouchAlignment Property

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.TouchAlignment</i> = {True   False}
<b>Description</b>	The <b>TouchAlignment</b> property lets you find or set whether touch alignment is turned on. The <b>TouchAlignment</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True turns touch alignment on; False turns it off.
<b>ABC Equivalent</b>	The <b>TouchAlignment</b> property is equivalent to choosing Preferences in the ABC File menu, clicking the Alignment button, and selecting the Touch Alignment option (True) or deselecting the option (False).

## Related Topics

### Language Elements

[AlignToRulers Property](#)  
[ChannelAlignment Property](#)  
[ShowRulers Property](#)  
[SmartShapeSpacing Property](#)  
[SSSHorizontal Property](#)  
[SSSVERTICAL Property](#)

### Description

[Alignment Options](#)

### Object

[Preferences Object](#)

## Units Property (Preferences Object)

Related

Related

Related

<b>Usage</b>	<i>PreferencesObject.Units = UnitsIndicator</i>						
<b>Description</b>	You use the <b>Units</b> property of the Preferences object to set whether positions are measured in inches or centimeters. The default is inches. The <b>Units</b> property is read/write.						
<b>Data Type</b>	Integer						
<b>Value</b>	The units used for measurements are listed in the table below. <table><thead><tr><th><b>UnitsIndicator</b></th><th><b>Description</b></th></tr></thead><tbody><tr><td>0</td><td>Inches</td></tr><tr><td>1</td><td>Centimeters</td></tr></tbody></table>	<b>UnitsIndicator</b>	<b>Description</b>	0	Inches	1	Centimeters
<b>UnitsIndicator</b>	<b>Description</b>						
0	Inches						
1	Centimeters						
<b>ABC Equivalent</b>	The <b>Units</b> property is equivalent to choosing Page Layout in the File menu and selecting the Inches option or the Centimeters option.						

## **Related Topics**

### **Language Elements**

[Units Property \(Chart Object\)](#)

### **Description**

[Drawing Unconnected Lines](#)

[Creating Text Blocks](#)

### **Object**

[Preferences Object](#)

## Shape Object

**Description** The Shape object is below the Object object. You can have only one Shape object for each Object object.

### Properties

---

[Application](#)  
[BorderColor](#)  
[BorderStyle](#)  
[BorderWidth](#)  
[FillColor](#)  
[FillPattern](#)  
[IsLaunched](#)  
[IsLinked](#)  
[LaunchCommand](#)  
[LaunchFlags](#)  
[LaunchStartDir](#)  
[LinkedChartName](#)  
[LinkFields](#)  
[NoteFont](#)  
[NoteText](#)  
[NoteTextLF](#)  
[Number](#)  
[NumberShown](#)  
[Parent](#)  
[ShadowColor](#)  
[ShadowOffset](#)  
[ShadowStyle](#)  
[ShapeName](#)

### Methods

---

[DeleteLines](#)  
[FitShapeToText](#)  
[Launch](#)  
[Link](#)  
[ReNumber](#)  
[ReplaceShape](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## BorderColor Property

Related

Related

Related

### Usage

*ShapeObject*.**BorderColor** = *Color*

### Description

You use the **BorderColor** property to set the border color for shapes using the **MakeRGB** method. A shape border includes not only the outside part of the shape, but also any interior lines used in the shape. For example, it includes the concentric circles on the inside of a 5.25" floppy disk shape. The **BorderColor** property is read/write.

### Data Type

Long

### Value

The color of the shape border

### ABC Equivalent

The **BorderColor** property is equivalent to selecting a shape, clicking the Shape tool in the toolbox, clicking the Apply button next to the Border Style button in the ribbon, and then clicking the color you want.

## **Related Topics**

### **Language Elements**

[BasicColor Method](#)

[Color Property \(Object Object\)](#)

[FillColor Property](#)

[MakeRGB Method](#)

[ShadowColor Property](#)

### **Description**

[Setting Shape Colors](#)

[Fill, Border, and Shadow Colors](#)

### **Object**

[Shape Object](#)

## Related

# BorderColor, BorderStyle, BorderWidth, FillPattern, FillColor Properties Example

This example uses the **BorderColor** property, **BorderStyle** property, **BorderWidth** property, **FillPattern** property, and **FillColor** property of the Shape object to set the border and fill of a shape.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim NewShapel As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set NewShapel = Chart.DrawShape("Decision")            ' Create a Decision shape
NewShapel.Shape.BorderColor = ABC.MakeRGB(0, 0, 255)   ' Make the border blue
NewShapel.Shape.BorderStyle = 1                       ' Make the border a solid line
NewShapel.Shape.BorderWidth = 3                      ' Give the border a medium width
NewShapel.Shape.FillPattern = 23                     ' Fill the shape with a brick pattern
NewShapel.Shape.FillColor = ABC.MakeRGB(255, 255, 0)  ' Fill the shape with yellow
```

## BorderStyle Property

Related

Related

Related

### Usage

*ShapeObject*.**BorderStyle** = *StyleNumber*

### Description

You use the **BorderStyle** property to find or set the line style for shape borders. A shape border includes not only the outside edge of a shape, but also any interior lines used in the shape (for example, the concentric circles on the inside of a 5.25" floppy disk shape). ABC provides many useful border styles, including solid and dashed lines and an invisible border. The **BorderStyle** property is read/write.

### Data Type

Integer

### Value

Set the **BorderStyle** property to 0 for an invisible border and to 1 for a solid border. The following illustration shows the values of the **BorderStyle** property for each available style.



### ABC Equivalent

The **BorderColor** property is equivalent to clicking a shape, clicking the Shape tool in the toolbox, clicking the Border Style button, and then clicking the border style you want.

## Related Topics

### Description

[Border Style and Width](#)

### Object

[Shape Object](#)

## BorderWidth Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.BorderWidth = WidthValue</i>
<b>Description</b>	You use the <b>BorderWidth</b> property to find or set the width of the border of a shape. The <b>BorderWidth</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The <b>BorderWidth</b> property can have a value ranging from 1 to 5, with 1 a hairline (the thinnest possible line) and 5 the thickest line width.
<b>ABC Equivalent</b>	The <b>BorderWidth</b> property is equivalent to clicking a shape, clicking the Shape tool in the toolbox, clicking the Apply button next to the border Style button in the ribbon, and then entering a border width.

## Related Topics

### Description

[Border Style and Width](#)

### Object

[Shape Object](#)

## FillColor Property

Related

Related

Related

Related

### Usage

*ShapeObject.FillColor = Color*

### Description

The **FillColor** property lets you find or set the fill color ( interior color) for shapes (see the **MakeRGB** method). The **FillColor** property is read/write.

### Data Type

Long

### Value

The fill color of the shape

### ABC Equivalent

The **FillColor** property is equivalent to selecting a shape, clicking the Shape tool in the tool box, clicking the Apply button next to the Fill Pattern button in the ribbon, and then clicking the color you want.

## Related Topics

### Language Elements

[BasicColor Method](#)

[BorderColor Property](#)

[Color Property \(Object Object\)](#)

[MakeRGB Method](#)

[ShadowColor Property](#)

### Description

[Setting Shape Colors](#)

[Fill, Border, and Shadow Colors](#)

### Object

[Shape Object](#)

## FillPattern Property

Related

Related

Related

### Usage

*ShapeObject.FillPattern = PatternNumber*

### Description

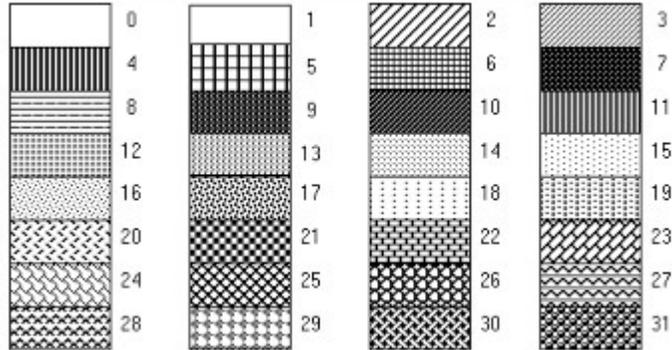
The **FillPattern** property lets you find or set a shape's fill pattern. The **FillPattern** property is read/write.

### Data Type

Integer

### Value

Set the **FillPattern** property to 0 for a transparent fill and to 1 for a solid fill. The following illustrations show the values of **FillPattern** for each available pattern.



### ABC Equivalent

The **FillPattern** property is equivalent to clicking a shape, clicking the Shape tool in the toolbox, clicking the Fill Pattern button, and then clicking the fill pattern you want.

## Related Topics

### Description

[Fill Pattern](#)

### Object

[Shape Object](#)

## IsLaunched Property

Related

Related

Related

**Usage** *ShapeObject.IsLaunched*

**Description** You can arrange to launch a program using a shape. You use the **IsLaunched** property to find if a shape has an associated program that it can launch. The **IsLaunched** property is read only.

**Data Type** Integer (Boolean)

**Value** True means the shape has an associated launch; False means it does not.

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[Launch Method](#)

### **Description**

[Launching Applications](#)

### **Object**

[Shape Object](#)

## IsLinked Property

Related

Related

Related

**Usage** *ShapeObject.IsLinked*

**Description** You can link shapes together. After the charts are linked, you can double click a designated shape in one chart to open the linked chart. You use the **IsLinked** property to determine if a shape is linked to another chart. The **IsLinked** property is read only.

**Data Type** Integer (Boolean)

**Value** True means the shape is linked; False means the shape is not linked.

**ABC Equivalent** None

## Related Topics

### Language Elements

[Link Method](#)

[LinkedChartName Property](#)

[LinkFields Property](#)

[LinkIndicator Property](#)

[LinkShadow Property](#)

### Description

[Linking Shapes to Other Charts](#)

### Object

[Shape Object](#)

## Number Property

Related

Related

Related

**Usage** *ShapeObject.Number = ShapeNumber*

**Description** You use the **Number** property to find or set the number of a shape. The **Number** property is read/write.

**Data Type** String

**Value** The number of a shape

**ABC Equivalent** None

## Related Topics

### Language Elements

[NextNumber Property](#)

[NumberShown Property](#)

[Renumber Method](#)

### Description

[Numbering Shapes](#)

### Object

[Shape Object](#)

## NumberShown Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.NumberShown</i> = {True   False}
<b>Description</b>	You use the <b>NumberShown</b> property to display or hide shape numbers. The <b>NumberShown</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the shape number is shown; False means it is not shown.
<b>ABC Equivalent</b>	The <b>NumberShown</b> property is equivalent to selecting a shape, clicking the Number tool in the toolbox, and clicking the Show/Hide Numbers button in the ribbon.

## Related Topics

### Language Elements

[NextNumber Property](#)

[Number Property](#)

[Renumber Method](#)

### Description

[Hiding Shape Numbers](#)

### Object

[Shape Object](#)

## LaunchFlags Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.LaunchFlags = Setting</i>
<b>Description</b>	You use the <b>LaunchFlags</b> property to find or set the flags for executing a launch. The <b>LaunchFlags</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The <b>LaunchFlags</b> property can take a combination of flags. The <i>Setting</i> is the sum of the values. The possible settings values for the flags are listed in the table below. <b>Setting Description</b> 1 Chart name 2 Shape number 4 Shape text
<b>ABC Equivalent</b>	The <b>LaunchFlags</b> property is equivalent to clicking the Object Selector tool in the toolbox, selecting the shape you want to use, clicking the Launch button in the ribbon, and specifying the flags in the Launch dialog box.

## **Related Topics**

### **Language Elements**

[Launch Method](#)

[LaunchCommand Property](#)

[LaunchIndicator Property](#)

[LaunchShadow Property](#)

[LaunchStartDir Property](#)

### **Description**

[Setting Shapes to Launch Applications](#)

### **Object**

[Shape Object](#)

## LaunchCommand Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject</i> . <b>LaunchCommand</b> = <i>Command</i> The <i>Command</i> element is the command that you want executed when the shape is launched, such as the path for the program to run.
<b>Description</b>	You use the <b>LaunchCommand</b> property to set a command to launch for the object. The LaunchCommand property is read/write.
<b>Data Type</b>	String
<b>Value</b>	The path and name of the program to run
<b>ABC Equivalent</b>	The <b>LaunchCommand</b> property is equivalent to clicking the Object Selector tool in the toolbox, selecting the shape you want to use, clicking the Launch button in the ribbon, and specifying the command line in the Launch dialog box.

## Related Topics

### Language Elements

[Launch Method](#)

[LaunchFlags Property](#)

[LaunchIndicator Property](#)

[LaunchShadow Property](#)

[LaunchStartDir Property](#)

### Description

[Setting Shapes to Launch Applications](#)

### Object

[Shape Object](#)

## LaunchStartDir Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.LaunchStartDir = Directory</i>
<b>Description</b>	You use the <b>LaunchStartDir</b> property to find or set the starting directory for a launch. The <b>LaunchStartDir</b> property is read/write.
<b>Data Type</b>	String
<b>Value</b>	The directory to be used for the launch
<b>ABC Equivalent</b>	The <b>LaunchStartDir</b> property is equivalent to clicking the Object Selector tool in the toolbox, selecting the shape you want to use, clicking the Launch button in the ribbon, and entering a directory in the Launch dialog box.

## Related Topics

### Language Elements

[Launch Method](#)

[LaunchCommand Property](#)

[LaunchFlags Property](#)

[LaunchIndicator Property](#)

[LaunchShadow Property](#)

### Description

[Setting Shapes to Launch Applications](#)

### Object

[Shape Object](#)

## LinkedChartName Property

Related

Related

Related

### Usage

*ShapeObject.LinkedChartName = ChartName*

### Description

You use the **LinkedChartName** property to provide a full pathname and filename for a chart and link the shape to the chart. The **LinkedChartName** property is read/write.

### Data Type

String

### Value

The full pathname and filename of the chart linked to the object

### ABC Equivalent

The **LinkedChartName** property is equivalent to clicking the Object Selector tool in the toolbox, selecting the shape you want to use, clicking the Link button in the ribbon, and entering the pathname in the Link dialog box.

## Related Topics

### Language Elements

[IsLinked Property](#)

[Link Method](#)

[LinkFields Property](#)

[LinkIndicator Property](#)

[LinkShadow Property](#)

### Description

[Linking Shapes to Other Charts](#)

### Object

[Shape Object](#)

## LinkFields Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.LinkFields</i> = {True   False}
<b>Description</b>	The <b>LinkFields</b> property lets you choose whether to accumulate the linked chart's field data as the object's field information if this object is linked to another chart with field information. The <b>LinkFields</b> property is read/write.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True accumulates the linked chart's field data as the object's field information; False does not accumulate it.
<b>ABC Equivalent</b>	The <b>LinkFields</b> property is equivalent to clicking the Object Selector tool in the toolbox, selecting the shape you want to use, clicking the Link button in the ribbon, and selecting or deselecting the Link Fields option.

## Related Topics

### Language Elements

[IsLinked Property](#)

[Link Method](#)

[LinkedChartName Property](#)

[LinkIndicator Property](#)

[LinkNOTIFY Event](#)

[LinkShadow Property](#)

[UpdateFields Method](#)

### Description

[Linking Shapes to Other Charts](#)

[Using Linked Field Data](#)

### Object

[Shape Object](#)

## NoteFont Property

Related

Related

Related

**Usage** *ShapeObject.NoteFont*

**Description** The **NoteFont** property lets you find the font object for notes. The **NoteFont** property is read only, but all the properties from the object it returns are read/write.

**Data Type** Object

**Value** The Font object for notes

**ABC Equivalent** None

## **Related Topics**

### **Language Elements**

[Font Property](#)

[NoteText Property](#)

[NoteTextLF Property](#)

### **Description**

[Text Typeface and Size](#)

### **Object**

[Shape Object](#)

## NoteText Property

Related

Related

Related

### Usage

*ShapeObject.NoteText = Note*

### Description

You use the **NoteText** property to find or set notes for a shape. You do not need to open the Note window to attach notes to a shape. If you wish, however, you can show or hide the ABC Note Viewer using the **NoteViewerVisible** property. You set the font for notes using the **NoteFont** property. If you wish to preserve Returns when reading a note, you should use the **NoteTextLF** property. The **NoteText** property is read/write.

### Data Type

String

### Value

The note associated with a shape

### ABC Equivalent

The **NoteText** property is equivalent to selecting a shape and entering text in the Note window.

## Related Topics

### Language Elements

[NoteFont Property](#)

[NoteIndicator Property](#)

[NoteShadow Property](#)

[NoteTextLF](#)

[NoteViewerVisible Property](#)

### Description

[Attaching a Note to a Shape](#)

[Adding Notes to a Shape](#)

### Object

[Shape Object](#)

## Related

### NoteText, NoteFont Properties Example

This example uses the **NoteText** property and **NoteFont** property of the Shape object to enter note text for a shape and change the text to red.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set Shape1 = Chart.DrawShape("Storage")                 ' Draw a Storage shape
Shape1.Shape.NoteText = ("Shape #" + Shape1.Shape.Number + " is a " + Shape1.Shape.ShapeName + "
shape.")        ' Enter text for the shape's note
Shape1.Shape.NoteFont.Color = ABC.MakeRGB(255, 0, 0)    ' Make the note text red
```

## ShadowColor Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.ShadowColor = Color</i>
<b>Description</b>	The <b>ShadowColor</b> property lets you find or set the shadow color for shapes (see the <b>MakeRGB</b> method). The <b>ShadowColor</b> property is read/write.
<b>Data Type</b>	Long
<b>Value</b>	The shadow color for the shape
<b>ABC Equivalent</b>	The <b>ShadowColor</b> property is equivalent to selecting a shape, clicking the Shape tool in the toolbox, clicking the Apply button next to the Shadow button in the ribbon, and then clicking the color you want.

## **Related Topics**

### **Language Elements**

[BasicColor Method](#)

[BorderColor Property](#)

[Color Property \(Object Object\)](#)

[FillColor Property](#)

[MakeRGB Method](#)

### **Description**

[Setting Shape Colors](#)

[Fill, Border, and Shadow Colors](#)

### **Object**

[Shape Object](#)

## ShadowOffset Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.ShadowOffset = OffsetAmount</i>
<b>Description</b>	You use the <b>ShadowOffset</b> property to find or set the width of a shadow (the distance the shadow appears away from its shape). The <b>ShadowOffset</b> property is read/write.
<b>Data Type</b>	Integer
<b>Value</b>	The <b>ShadowOffset</b> property can have a value ranging from 1 to 5, with 1 a hairline (the thinnest possible shadow) and 5 the thickest shadow.
<b>ABC Equivalent</b>	The <b>ShadowOffset</b> property is equivalent to clicking a shape, clicking the Shape tool in the toolbox, clicking the Apply button next to the Shadow Style button in the ribbon, and then entering a shadow width.

## Related Topics

### Language Elements

[ShadowStyle Property](#)

### Description

[Shadow Style and Width](#)

### Object

[Shape Object](#)

## ShadowStyle Property

Related

Related

Related

### Usage

*ShapeObject.ShadowStyle = ShadowStyleNumber*

### Description

You use the **ShadowStyle** property to find or set the position of the shadow on a shape. The **ShadowStyle** property is read/write.

### Data Type

Integer

### Value

**ShadowStyle** can have a value from 0 to 4, with 0 being no shadow and 1 through 4 the positions shown in the following illustration.



### ABC Equivalent

The **ShadowStyle** property is equivalent to clicking a shape, clicking the Shape tool in the toolbox, clicking the Shadow Style button, and then clicking the shadow style you want.

## Related Topics

### Language Elements

[ShadowOffset Property](#)

### Description

[Shadow Style and Width](#)

### Object

[Shape Object](#)

## ShapeName Property

Related

Related

Related

<b>Usage</b>	<i>ShapeObject</i> . <b>ShapeName</b>
<b>Description</b>	The <b>ShapeName</b> property lets you find the name of a shape, such as "Process" or "Decision." The <b>ShapeName</b> property is read only.
<b>Data Type</b>	String
<b>Value</b>	The name of the shape
<b>ABC Equivalent</b>	None

## **Related Topics**

### **Language Elements**

[Type Property \(Object object\)](#)

[UniqueID Property](#)

### **Description**

[Identifying an Object](#)

[Selecting Shapes](#)

### **Object**

[Shape Object](#)

## Related

# ShapeName, ShadowStyle, ShadowColor, ShadowOffset Properties Example

This example uses the **ShapeName** property, **ShadowStyle** property, **ShadowColor** property, and **ShadowOffset** property of the Shape object to identify a shape and apply a shadow in the desired location, color, and offset. To show the proper result, ABC should be open and contain shapes, at least one of which is a Decision shape.

```
Dim ABC As Object, Chart As Object, Shape As Object
Dim CurrentShape As Object
Dim SelectedShapes As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set SelectedShapes = Chart.Objects

Do
    Set CurrentShape = SelectedShapes.ItemFromShapes    ' Check all shapes in the chart
    If CurrentShape.Shape.ShapeName = "Decision" Then ' If a shape is a Decision
        CurrentShape.Shape.ShadowStyle = 1            ' Apply shadow to bottom right
        CurrentShape.Shape.ShadowColor = ABC.MakeRGB(0, 0, 255) ' Make shadow blue
        CurrentShape.Shape.ShadowOffset = 5           ' Give the shadow a big offset
    End If
Loop While CurrentShape.Valid
```

## DeleteLines Method

Related

Related

Related

### Usage

*ShapeObject.DeleteLines*

### Description

You use the **DeleteLines** method of the Shape object to delete all the lines attached to a specific shape. Deleting lines with this method does not place the lines in the Windows Clipboard.

**ABC Equivalent** None

## **Related Topics**

### **Description**

[Deleting Lines](#)

[Cutting, Copying, Pasting, Duplicating, and Clearing Objects](#)

### **Object**

[Shape Object](#)



## FitShapeToText Method

Related

Related

Related

**Usage** *ShapeObject.FitShapeToText*

**Description** You use the **FitShapeToText** method to expand or contract a shape around its center so that the text in it fits. This is useful when the length of the text string may vary and you want to avoid hiding text that will not fit within the shape.

**ABC Equivalent** The **FitShapeToText** method is equivalent to selecting the shape, clicking the Text tool, and then clicking the Fit Shape button in the ribbon.

## Related Topics

### Description

[Fitting Shapes to Text](#)

[Sizing Shapes to Text](#)

### Object

[Shape Object](#)



## Launch Method

Related

Related

Related

**Usage** *ShapeObject.Launch*

**Description** You use the **Launch** method to execute the shape's launch.

**Data Type** Integer (Boolean)

**Value** True means the launch was successful; False means the launch was not successful.

**ABC Equivalent** The **Launch** method is equivalent to pressing **Ctrl** and double clicking the shape you set for launching

## Related Topics

### Language Elements

[LaunchCommand Property](#)

[LaunchFlags Property](#)

[LaunchIndicator Property](#)

[LaunchShadow Property](#)

[LaunchStartDir Property](#)

### Description

[Launching Applications](#)

### Object

[Shape Object](#)

## Related

### Launch Example

This example uses the **IsLaunched** property, **LaunchFlags** property, **LaunchCommand** property, **LaunchStartDir** property, and **Launch** method of the Shape object to determine if a shape has a launch attached to it and, if it does not, to set launch flags, define the program to be launched, set the directory for the launched application, and launch the application.

```
Dim ABC As Object, Chart As Object
Dim Shape As Object
Dim CurrentShape As Object

Set ABC = CreateObject("ABCFlow.application")      ' Start ABC
ABC.Visible = True                                ' Make ABC visible
Set Chart = ABC.ActiveChart                        ' Get the active chart

Set CurrentShape = Chart.Objects.ItemFromShapes    ' Select a shape
If CurrentShape.Shape.IsLaunched = False Then      ' If shape is not launched, set up
    CurrentShape.Shape.LaunchFlags = 1            ' Pass the chart name
    CurrentShape.Shape.LaunchCommand = "c:\windows\notepad.exe" ' App to launch
    CurrentShape.Shape.LaunchStartDir = "c:\temp"  ' Launched application's file path
    CurrentShape.Shape.Launch                     ' Execute the launch
Else MsgBox "This shape already has a launch."
End If
```

## Link Method

Related

Related

Related

<b>Usage</b>	<i>ShapeObject.Link</i>
<b>Description</b>	You use the <b>Link</b> method to open the chart linked to a shape. If you have not yet set a value for the <b>LinkedChartName</b> property, this method creates a new chart with an automatically generated filename using the <b>DefaultFilePath</b> property.
<b>Data Type</b>	Object
<b>Value</b>	The linked chart
<b>ABC Equivalent</b>	The <b>Link</b> method is equivalent to double-clicking the linked shape. The linked chart opens and becomes the active chart.

## Related Topics

### Language Elements

[DefaultFilePath Property](#)

[GroupAndLink Method](#)

[IsLinked Property](#)

[LinkedChartName Property](#)

[LinkFields Property](#)

[LinkShadow Property](#)

### Description

[Opening a Linked Chart](#)

### Object

[Shape Object](#)

## Related

### Link Method, LinkFields Property, and LinkShadow Property Example

This example uses the **Link** method and the **LinkFields** property of the Shape object and the **LinkShadow** property of the Chart object to link a shape to show a shadow on linked shapes, link a shape to another chart, and deselect Link Fields in the Link dialog box.

```
Dim ABC As Object, Chart As Object
Dim Shapel As Object, Link1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
ABC.CloseAll
ABC.New
Set Chart = ABC.ActiveChart

Chart.LinkShadow = True
Set Shapel = Chart.DrawShape("Decision")
Set Link1 = Shapel.Shape.Link
ABC.TileCharts

Shapel.Shape.LinkFields = 0
MsgBox "Accumulations of fields from the linked chart will not appear in the top chart."
```

```
' Start ABC
' Make ABC visible
' Close all charts
' Add a new chart
' Get the active chart

' Show a shadow on linked shapes
' Draw a shape
' Link the shape to a new chart
' Tile all chart windows

' Deselect Link Fields in the Link dialog
box
```

## Renumber Method

Related

Related

Related

**Usage** *ShapeObject*.**Renumber**

**Description** You use the **Renumber** method to replace the current shape number with the value in the **NextNumber** property and increment the value in the **NextNumber** property.

**ABC Equivalent** The **Renumber** method is equivalent to selecting a shape, clicking the Number tool in the toolbox, and clicking the Renumber button in the ribbon.

## Related Topics

### Language Elements

[NextNumber Property](#)

[Number Property](#)

[NumberShown Property](#)

### Description

[Numbering Shapes](#)

### Object

[Shape Object](#)

## ReplaceShape Method

Related

Related

Related

<b>Usage</b>	<i>ShapeObject</i> . <b>ReplaceShape</b> [ <i>ShapeName</i> ] The <i>ShapeName</i> element is the name of the shape to put in place of the selected shape.
<b>Description</b>	You use the <b>ReplaceShape</b> method of the Shape object to replace shapes. The new shape connects to the lines of the old shape. You can replace shapes with the chart's current shape or can specify a shape.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the shape was replaced successfully; False means the replacement was not successful. The <i>ShapeName</i> element is a string.
<b>ABC Equivalent</b>	The <b>ReplaceShape</b> method is equivalent to selecting a shape, choosing the new shape in the Shape Palette, then clicking the Shape tool in the toolbox, and clicking the Replace Shape button in the ribbon.

## Related Topics

### Language Elements

[DrawShape Method](#)

### Description

[Replacing Shapes](#)

### Object

[Shape Object](#)

**Related**

## ReplaceShape Method Example

This example uses the **ReplaceShape** method of the Shape object to replace one shape with another.

```
Dim ABC As Object, Chart As Object
Dim Shape1 As Object

Set ABC = CreateObject("ABCFlow.application")
ABC.Visible = True
Set Chart = ABC.ActiveChart

Set Shape1 = Chart.DrawShape("Terminal")
Shape1.Shape.ReplaceShape "Delay"
```

' Start ABC  
' Make ABC visible  
' Get the active chart  
  
' Draw a Terminal shape  
' Change Shape1 into a Delay shape

## NoteTextLF Property

Related

Related

Related

### Usage

*ShapeObject.NoteTextLF = Note*

### Description

You use the **NoteTextLF** property to find or set notes for a shape. When adding a note, the property is identical to the **NoteText** property. When reading the text in a note, the property does not substitute spaces for Returns as the **NoteText** property does. If you do not wish to preserve Returns, you should use the **NoteText** property.

You do not need to open the Note window to attach notes to a shape. If you wish, however, you can show or hide the ABC Note Viewer using the **NoteViewerVisible** property. You set the font for notes using the **NoteTextLF** property. The **NoteTextLF** property is read/write.

### Data Type

String

### Value

The note associated with a shape with Returns preserved

### ABC Equivalent

None

## **Related Topics**

### **Language Elements**

[NoteFont Property](#)

[NoteIndicator Property](#)

[NoteShadow Property](#)

[NoteText](#)

[NoteViewerVisible Property](#)

### **Description**

[Attaching a Note to a Shape](#)

[Adding Notes to a Shape](#)

### **Object**

[Shape Object](#)



## TextBlock Object

**Description** The TextBlock object is below the Object object. You can have only one TextBlock object for each Object object.

### Properties

[Application](#)  
[AttachedToLine](#)  
[Parent](#)

### Methods

[UnattachFromLine](#)

## Related Topics

### Description

[ABC Object Hierarchy](#)

[Language reference](#)

[Objects, alphabetical](#)

[Objects, graphical](#)

[Properties and Methods by Task](#)

## AttachedToLine Property

Related

Related

Related

<b>Usage</b>	<i>TextBlockObject.AttachedToLine</i>
<b>Description</b>	The <b>AttachedToLine</b> method lets you find whether a text block is attached to a line. The <b>AttachedToLine</b> method is read only.
<b>Data Type</b>	Integer (Boolean)
<b>Value</b>	True means the text block is attached to a line; False means it is not.
<b>ABC Equivalent</b>	None

## Related Topics

### Language Elements

[TextBlock Property](#)

[UnattachFromLine Method](#)

### Description

[Unattaching Text from a Line](#)

### Object

[TextBlock Object](#)

## UnattachFromLine Method

Related

Related

Related

**Usage** *TextBlockObject*.**UnattachFromLine**

**Description** The **UnattachFromLine** method lets you detach a text block from all lines.

**ABC Equivalent** The **UnattachFromLine** method is equivalent to dragging a text block away from lines to detach it from the lines.

## Related Topics

### Language Elements

[AttachedToLine Property](#)

### Description

[Unattaching Text from a Line](#)

### Object

[TextBlock Object](#)

**Related**

## UnattachFromLine Method Example

This example uses the **UnattachFromLine** method of the **TextBlock** object to detach text from a line.

```
Dim ABC As Object, Chart As Object
Dim NewLine As Object
Dim NewText As Object

Set ABC = CreateObject("ABCFlow.application")           ' Start ABC
ABC.Visible = True                                     ' Make ABC visible
Set Chart = ABC.ActiveChart                            ' Get the active chart

Set NewLine = Chart.DrawFreeLine(4,4)                  ' Draw a line
Set NewText = Chart.DrawTextBlock("Hello there!")     ' Draw a text block
NewLine.Line_.AttachText NewText                       ' Attach the text to the line
MsgBox "The text is attached to the line. When you click OK, it will be detached."
NewText.TextBlock.UnattachFromLine                    ' Detach text from the line
```

## ABC Events

AppQuitNOTIFY  
AppQuitSUBCLASS  
AppMenuSUBCLASS  
AppMenuHintSUBCLASS  
AppMenuPopupSUBCLASS  
ChartActivateNOTIFY  
ChartChangeNOTIFY  
ChartCloseSUBCLASS  
ChartNewNOTIFY  
ChartOpenNOTIFY  
ChartPasteNOTIFY  
DeleteSUBCLASS  
DoubleClickSUBCLASS  
ExclusiveSelectionNOTIFY  
FieldValueChangedNOTIFY  
LinkNOTIFY  
NewLineNOTIFY  
NewShapeNOTIFY  
ObjectClickSUBCLASS  
ObjectFontChangeNOTIFY  
ObjectLineAttachNOTIFY  
ObjectMovedNOTIFY  
ObjectMoveSUBCLASS  
ObjectSizedNOTIFY  
ObjectSizeSUBCLASS  
ObjectTextChangedNOTIFY  
ReplaceShapeNOTIFY  
SpecialKeySUBCLASS

## Related Topics

### Language Elements

[ChartTypeShutdown](#)

[RegisterEvent](#)

[UnRegisterEvent](#)

### Description

[Language reference](#)

## AppQuitNOTIFY Event

Related

**Description** The **AppQuitNOTIFY** event occurs when ABC is closed. The **AppQuitNOTIFY** event procedure can be used for final actions that you want your program to perform before it closes. If you want the Visual Basic application to close when ABC does, put a Visual Basic End statement in this procedure.

## Related Topics

### Description

[When ABC Closes](#)

### Object

[ABC Events](#)

## AppQuitSUBCLASS Event

Related

### Description

The **AppQuitSUBCLASS** event occurs when a request is made to close ABC. The user can request that ABC close by choosing Exit in the ABC File menu, pressing **Alt+F4**, or double clicking the ABC window Control box. The **AppQuitSUBCLASS** event procedure is triggered before ABC closes. You can prevent ABC from closing by setting the ABC1 object **Override** property to True.

## Related Topics

### Description

[When ABC Closes](#)

### Object

[ABC Events](#)

## AppMenuSUBCLASS Event

Related

**Description** The **AppMenuSUBCLASS** event occurs when the user chooses an item on an add-on menu. The menu item object that was chosen is passed to the event procedure in the **MenuItem** variable.

## Related Topics

### Description

[When Menultems Are Chosen](#)

### Object

[ABC Events](#)

## AppMenuHintSUBCLASS Event

Related

**Description** The **AppMenuHintSUBCLASS** event occurs when the user moves the menu cursor to an item on an add-on menu. The **AppMenuHintSUBCLASS** event procedure is triggered before ABC highlights the menu item. The menu item object to be highlighted is passed to the event procedure in the MenuItem variable.

## Related Topics

### Description

[When Menultems Are Highlighted](#)

### Object

[ABC Events](#)

## AppMenuPopupSUBCLASS Event

Related

Related

### Description

The **AppMenuPopupSUBCLASS** event occurs when the user opens an add-on menu by clicking the menu's name. Add-on menus are created with the **AddMenu** method of the Application object. The **AppMenuPopupSUBCLASS** event procedure is triggered before ABC displays the add-on menu. The Menu object about to open is passed to the event procedure in the Menu variable.

Because the **AppMenuPopupSUBCLASS** event is triggered before the add-on menu opens, you can use this event procedure to determine whether any items on the add-on menu should be disabled (gray) or checked. A menu item is disabled by setting the **Enabled** property of the MenuItem object to False. A menu item is checked by setting the **Checked** property of the MenuItem object to True.

## Related Topics

### Description

[When Add-On Menus Open](#)

### Object

[ABC Events](#)

## ChartActivateNOTIFY Event

Related

### Description

The **ChartActivateNOTIFY** event occurs when a chart is activated when a user clicks on it, chooses it from the Window menu or the Chart's System Menu Next, etc. The **ChartActivateNOTIFY** event procedure is triggered following the activation of the chart. The activated chart object is passed to the event procedure in the Chart object variable.

## Related Topics

### Description

[When Charts Are Activated](#)

### Object

[ABC Events](#)

## ChartChangeNOTIFY Event

Related

### Description

The **ChartChangeNOTIFY** event occurs when a chart is changed in any way. The **ChartChangeNOTIFY** event procedure is triggered following the changing of the chart. The changed chart object is passed to the event procedure in the Chart object variable.

## Related Topics

### Description

[When Charts Change](#)

### Object

[ABC Events](#)

## ChartCloseSUBCLASS Event

Related

**Description** The **ChartCloseSUBCLASS** event occurs when the user closes a chart by choosing Close in the ABC File menu. The **ChartCloseSUBCLASS** event procedure is triggered before the closing of the chart. The chart object that is about to close is passed to the event procedure in the Chart object variable.

## Related Topics

### Description

[When Charts Close](#)

### Object

[ABC Events](#)

## ChartNewNOTIFY Event

Related

**Description** The **ChartNewNOTIFY** event occurs when the user creates a new chart by choosing New in the File menu of ABC. The **ChartNewNOTIFY** event procedure is triggered following the creation of the new chart. The new chart object is passed to the event procedure in the Chart object variable.

## Related Topics

### Description

[When New Charts Are Created](#)

### Object

[ABC Events](#)

## ChartOpenNOTIFY Event Related

**Description** The **ChartOpenNOTIFY** event occurs when the user opens a new chart file by choosing Open in the File menu of ABC. The **ChartOpenNOTIFY** event procedure is triggered following the successful opening of the chart file. The opened chart object is passed to the event procedure in the Chart object variable.

## Related Topics

### Description

[When Charts Open](#)

### Object

[ABC Events](#)

## ChartPasteNOTIFY Event Related

**Description** The **ChartPasteNOTIFY** event occurs when a user pastes something into a chart by choosing Paste in the Edit menu of ABC. The **ChartPasteNOTIFY** event procedure is triggered following the paste. The chart object is passed to the event procedure in the Chart object variable.

## Related Topics

### Description

[When Charts Are Pasted](#)

### Object

[ABC Events](#)

## DeleteSUBCLASS Event Related

**Description** The **DeleteSUBCLASS** event occurs when one or more Objects are deleted. The user deletes Objects by selecting the Objects and pressing **del** or choosing Clear from the Edit menu. The **DeleteSUBCLASS** event procedure is triggered before ABC performs the deletion. The Object to be deleted first is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Objects Are Deleted](#)

### Object

[ABC Events](#)

## DoubleClickSUBCLASS Event

Related

### Description

The **DoubleClickSUBCLASS** event occurs when the user double clicks a Shape object. The **DoubleClickSUBCLASS** event procedure is triggered before ABC shows the Shape as selected. The clicked Shape is passed to the event procedure in the Object variable, and the chart in which the shape is located is passed in the Chart variable.

## Related Topics

### Description

[When Shapes Are Double Clicked](#)

### Object

[ABC Events](#)

## ExclusiveSelectionNOTIFY Event

Related

### Description

The **ExclusiveSelectionNOTIFY** event occurs when the user selects a single Object object. The **ExclusiveSelectionNOTIFY** event procedure is triggered after ABC shows the Object as selected. The selected Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable. **Note:** If the user selects more than one object, the **ExclusiveSelectionNOTIFY** event is not activated.

## Related Topics

### Description

[When Objects Are Selected](#)

### Object

[ABC Events](#)

## FieldValueChangedNOTIFY Event

Related

### Description

The **FieldValueChangedNOTIFY** event occurs when the user changes a FieldValue object. The **FieldValueChangedNOTIFY** event procedure is triggered after ABC changes the FieldValue. The FieldValue that was changed is passed to the event procedure in the FieldValue variable, the Object that owns the field is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Field Values Change](#)

### Object

[ABC Events](#)

## LinkNOTIFY Event

Related

### Description

The **LinkNOTIFY** event occurs when a chart file is opened by double clicking the object to which it is linked. The **LinkNOTIFY** event procedure is triggered following the successful opening of the chart file. The chart object from which the linked chart was opened (the source chart) is passed to the event procedure in the Chart object variable. The linked chart object (the chart just opened) can be obtained using the **ActiveChart** property of the Application object. The Object that was double clicked in the source chart to open the linked chart is passed to the event procedure in the Object object variable.

## Related Topics

### Language Elements

[IsLinked Property](#)

[Link Method](#)

### Description

[When Linked Charts Open](#)

### Object

[ABC Events](#)

## NewLineNOTIFY Event

Related

### Description

The **NewLineNOTIFY** event occurs when the user draws a new Line object. The **NewLineNOTIFY** event procedure is triggered after ABC draws the Line. The drawn Line is passed to the event procedure in the Object variable, and the chart in which the Line is located is passed in the Chart variable.

## Related Topics

### Description

[When Lines Are Drawn](#)

### Object

[ABC Events](#)

## NewShapeNOTIFY Event

**Description** The **NewShapeNOTIFY** event occurs when the user draws a new Shape object. The **NewShapeNOTIFY** event procedure is triggered after ABC draws the Shape. The drawn Shape is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

## Related Topics

### Description

[When Shapes Are Drawn](#)

### Object

[ABC Events](#)

## ObjectClickSUBCLASS Event

Related

### Description

The **ObjectClickSUBCLASS** event occurs when the user clicks an object. The **ObjectClickSUBCLASS** event procedure is triggered before ABC shows the Object as selected. The clicked Object is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Objects Are Clicked](#)

### Object

[ABC Events](#)

## ObjectFontChangeNOTIFY Event

Related

**Description** The **ObjectFontChangeNOTIFY** event occurs when the user changes the font of one or more Text objects. The **ObjectFontChangeNOTIFY** event procedure is triggered after ABC displays the Text objects in the changed font. The Text object that was changed first is passed to the event procedure in the Object variable, and the chart in which the text is located is passed in the Chart variable.

## Related Topics

### Description

[When Fonts Change](#)

### Object

[ABC Events](#)

## ObjectLineAttachNOTIFY Event

Related

### Description

The **ObjectLineAttachNOTIFY** event occurs when the user attaches a line to an Object. The **ObjectLineAttachNOTIFY** event procedure is triggered after ABC attaches the Line. The Object to which the line is attached is passed to the event procedure in the Object variable, the line is passed in the Object2 variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Lines Attach](#)

### Object

[ABC Events](#)

## ObjectMovedNOTIFY Event

Related

### Description

The **ObjectMovedNOTIFY** event occurs when an Object object is moved. The **ObjectMovedNOTIFY** event procedure is triggered after ABC has moved the Object. The Object that was moved is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Objects Move](#)

### Object

[ABC Events](#)

## ObjectMoveSUBCLASS Event

Related

### Description

The **ObjectMoveSUBCLASS** event occurs when the user starts to move an Object object. The **ObjectMoveSUBCLASS** event procedure is triggered before ABC initiates any move behavior. The Object about to move is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Objects Move](#)

### Object

[ABC Events](#)

## ObjectSizedNOTIFY Event

Related

### Description

The **ObjectSizedNOTIFY** event occurs when an Object object is resized. The **ObjectSizedNOTIFY** event procedure is triggered after ABC has resized the Object. The Object that was resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Objects Are Resized](#)

### Object

[ABC Events](#)

## ObjectSizeSUBCLASS Event Related

**Description** The **ObjectSizeSUBCLASS** event occurs when the user starts to resize an Object object. The **ObjectSizeSUBCLASS** event procedure is triggered before ABC initiates any resizing behavior. The Object to be resized is passed to the event procedure in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Objects Are Resized](#)

### Object

[ABC Events](#)

## ReplaceShapeNOTIFY Event

Related

### Description

The **ReplaceShapeNOTIFY** event occurs when the user replaces one or more Shape objects. The **ReplaceShapeNOTIFY** event procedure is triggered after ABC replaces the Shape objects. The Shape to be replaced first is passed to the event procedure in the Object variable, and the chart in which the Shape is located is passed in the Chart variable.

## Related Topics

### Description

[When Shapes Are Replaced](#)

### Object

[ABC Events](#)

## SpecialKeySUBCLASS Event

Related

**Description** The **SpecialKeySUBCLASS** event occurs when the user presses one of the special keys. The **SpecialKeySUBCLASS** event procedure is triggered before ABC responds to the key press. A code representing the key is passed to the event procedure in the **WParam** variable. These codes are defined in the table below.

<b>Key</b>	<b>Code</b>
F1	1
F2	2
F3	3
F4	4
F5	5
F6	6
F7	7
F8	8
F9	9
F10	10
F11	11
F12	12
Tab	13
Esc	27
PgUp	33
PgDn	34
End	35
Home	36
Left Arrow	37
Up Arrow	38
Right Arrow	39
Down Arrow	40
Ins	45
Del	46

## Related Topics

### Description

[When Special Keys Are Pressed](#)

### Object

[ABC Events](#)

## ObjectTextChangedNOTIFY Event

Related

**Description** The **ObjectTextChangedNOTIFY** event occurs when the user changes a text block. The **ObjectTextChangedNOTIFY** event procedure is triggered after ABC changes the TextBlock. The Object that owns the text is passed in the Object variable, and the chart in which the Object is located is passed in the Chart variable.

## Related Topics

### Description

[When Text Changes](#)

### Object

[ABC Events](#)

Related

## ABC Features Not Automated

Related

[Importing Files in Other Formats](#)

[Printing to a File](#)

[Saving a Workspace](#)

[Special Shape Properties](#)

[Editing Shapes in the Shape Palette](#)

[Setting Preferences for the Shape Palette](#)

[Printing Notes Without Shapes](#)

[Spacing and Aligning Shapes](#)

[The Index](#)

[Moving Guidelines](#)

[Opening ABC 1.x Files](#)

[Using Mixed Fonts](#)

[Drawing Certain Lines](#)

## Related Topics

Description

[Contents](#)

## Importing Files in Other Formats

Related

In ABC, you use the Import command in the File menu to import charts with other formats into ABC charts. This is not a common operation. To support it, there would have to be ABC OLE Automation commands for all the dialog boxes for naming the subcharts, choosing the chart font, and so forth.

If you need to import charts with other formats, you should have your user perform the command in ABC rather than using ABC OLE Automation.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Printing to a File

Related

In ABC you print to a file by choosing the Print command in the File menu. Then, in the Print dialog box, you choose the Print to File option before you print. Printing to a file can be convenient, but often you can print directly to a printer. Printing to a file may be supported in a later version of ABC OLE Automation.

With some printers, you can print to a file by setting an option in the driver, usually in the driver's Options dialog box. You always can print to a file by using the Windows Control Panel to set the driver to the port FILE:. If you need to print to a file and cannot do it using the printer driver, you should have your user perform the action in ABC rather than using ABC OLE Automation.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Saving a Workspace

Related

You usually use the Save Workspace command in the File menu when you have a standard method of working. With ABC OLE Automation, you can set up a standard appearance using the commands to open files and position their windows. (For more information on positioning chart windows, see [Running and Setting Up ABC.](#))

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Special Shape Properties Related

ABC OLE Automation does not let you set connect points, set the rectangle that contains text, or specify label alignment. These abilities are often used to make corrections due to the way that objects were originally created. With ABC OLE Automation, there is less likely a need for these changes.

If you have a particular need that is outside what ABC normally does, you can set up a template and use it as the basis for a chart using [AddFromTemplate](#). Alternatively, you can have your user perform the commands in ABC rather than using ABC OLE Automation.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Editing Shapes in the Shape Palette

Related

In ABC, you can choose Edit from the Palette menu and perform activities such as arranging the shapes in a palette, pasting new shapes into a palette, or deleting shapes from a palette. None of these activities are available in ABC OLE Automation. You also cannot alter the shape properties for a particular shape in the palette. If customized palettes are required, use ABC to set up the palettes and then reference the custom palette.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Setting Preferences for the Shape Palette

Related

In ABC, you can choose Preferences from the Palette menu and perform activities such as setting the palette title bar or button face size. These activities are not available in ABC OLE Automation. If customized palettes are required, use ABC to set up the palettes and then reference the custom palette.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Printing Notes Without Shapes

Related

In ABC you can print notes directly from the Note window by choosing the Print command from the Note menu. ABC OLE Automation does not allow printing only notes. To print notes, you must print both notes and their associated shapes (the equivalent of using the Print command in the File menu of the main window).

If you need to print notes without shapes, you should have your user perform the command in ABC rather than using ABC OLE Automation.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Spacing and Aligning Shapes

Related

To set the spacing and alignment in ABC, you select two or more shapes and then click the Space/Alignment button in the Selector ribbon. The Space/Align Shapes dialog box opens so you can specify options.

For almost all ABC OLE Automation applications, spacing and alignment are handled when you create the objects. If necessary, you can use the positioning commands to place objects where you want them. (For descriptions of the positioning commands, see [Working with Shapes](#) and [Language Reference](#).)

If necessary, you can have your user perform the alignment in spacing in ABC rather than using ABC OLE Automation.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## The Index

Related

You use the index in ABC to relate the current chart to other charts. With ABC OLE Automation, you can control the relationship to other charts using facilities such as the [LinkNOTIFY event](#); the [Link method](#); and the [LinkIndicator](#), [LinkShadow](#), [IsLaunched](#), [IsLinked](#), [LinkedChartName](#), and [LinkFields](#) properties. These are described in [Using Linked Field Data](#).

You can show the Index using the [IndexVisible property](#), which is described in the section [Displaying the Field Viewer, Notes Viewer, Index, and Shape Palette](#).

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Moving Guidelines

Related

Guidelines are a powerful feature of ABC. With ABC OLE Automation, you can create guidelines using the [AddHorizontalGuideline](#) and [AddVerticalGuideline](#) methods. You can choose whether guidelines display in the chart using the [GuidelinesOn](#) property. You can delete all guidelines using the [ClearGuidelines](#) method. These facilities are described in [Using Guidelines](#).

After you create guidelines, you cannot act on them except to delete all the guidelines in a chart. If you need to change the placement of guidelines, you should delete them all and recreate the ones you want in the new positions you need.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Opening ABC 1.x Files

Related

You cannot open ABC 1.x files using ABC OLE Automation.

If necessary, you can have your user open the files in ABC rather than using ABC OLE Automation.

## Related Topics

### Description

[ABC Facilities Not Supported](#)

## Using Mixed Fonts

Related

You can specify the font for a particular object. (See the [Working with Text](#) for more information.) There is no way to specify different fonts within a particular piece of text, however.

If necessary, you can have your user specify fonts for a particular word or phrase rather than using ABC OLE Automation.

## Related Topics

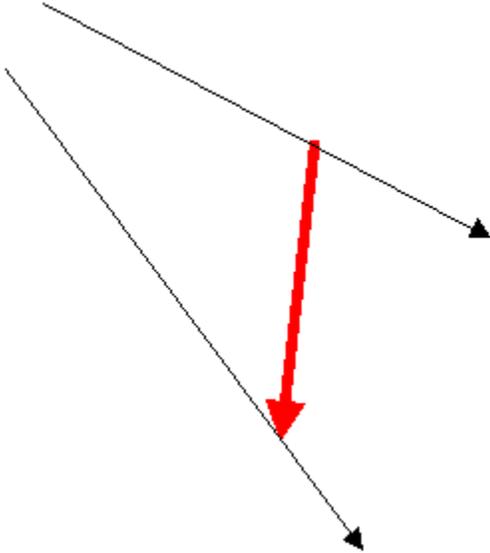
### Description

[ABC Facilities Not Supported](#)

## Drawing Certain Lines

Related

It is not possible to draw certain lines using ABC OLE Automation because there is no way to specify the start and finish positions of the line. For example, in the following illustration you cannot specify how to draw the red line.



If necessary, you can have your user draw the line rather than using ABC OLE Automation.

## Related Topics

### Description

[ABC Facilities Not Supported](#)



**Index**

## Properties and Methods by Task

Related

The related topics list tasks you can perform in ABC using OLE Automation, and the properties and methods applicable for each task.

Use these topics as a handy reference to the properties and methods associated with various tasks.

[Tasks: Setting up the Application](#)

[Tasks: Working with Chart Files](#)

[Tasks: Working with Objects](#)

[Tasks: Working with Shapes](#)

[Tasks: Working with Lines](#)

[Tasks: Working with Text](#)

[Tasks: Working with Data Fields](#)

[Tasks: Using Color](#)

[Tasks: Customizing Menus](#)

[Tasks: Special Programming Features](#)

## Related Topics

### Description

[Contents](#)

[Language Reference](#)

[Searching in Help](#)

## Tasks: Setting up the Application

Related

### Task Properties/Methods

Displaying the ABC window

[Visible](#), [Activate](#), [ArrangeIcons](#), [CascadeCharts](#), [TileCharts](#), [Maximize](#), [Minimize](#)

Resizing and positioning the window

[Bottom](#), [Height](#), [Left](#), [Right](#), [Top](#), [Width](#), [Restore](#)

Changing and reading the title bar

[Caption](#)

Changing and reading the status bar

[StatusBar](#), [Hint](#)

Displaying the field viewer, notes viewer, index, and shape palette

[FieldViewerVisible](#), [IndexVisible](#), [NoteViewerVisible](#), [ShapePaletteVisible](#)

Getting ABC system information

[FullName](#), [Name](#), [Path](#), [Version](#), [OperatingSystem](#)

Customizing preferences

[AlignToRulers](#), [ChannelAlignment](#), [LineSpacingX](#), [LineSpacingY](#), [ShapeSizing](#), [ShowRulers](#), [SSSHorizontal](#), [SSSVERTICAL](#), [SmartShapeSpacing](#), [TouchAlignment](#), [Preferences](#), [LinkIndicator](#), [LinkShadow](#), [LaunchIndicator](#), [LaunchShadow](#), [NoteIndicator](#), [NoteShadow](#), [ShowNodesOnLines](#), [NumberFont](#), [SetDefaults](#)

Displaying help

[Help](#)

Closing ABC

[Quit](#)

Choosing the target printer

[Printer](#)

Customizing the menu

[AddMenu](#), [RemoveMenu](#)

Customizing the toolbox

[CreateAddOn](#), [RemoveAddOn](#)

Setting the default path

[DefaultFilePath](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Working with Chart Files

Related

### Task Properties/Methods

Creating new charts

[New](#), [NewFromTemplate](#), [Add](#), [AddFromTemplate](#)

Opening charts

[Open](#), [DefaultFilePath](#)

Saving charts

[HasDiskFile](#), [ReadOnly](#), [Saved](#), [RevertToSaved](#), [Save](#), [DefaultFilePath](#)

Closing charts

[CloseChart](#), [CloseAll](#)

Activating a chart

[Item](#), [Count](#), [ActiveChart](#), [Name](#), [Activate](#), [Charts](#), [FullName](#)

Protecting charts

[Protected](#), [SetProtection](#)

Linking charts

[LinkedChartName](#), [IsLinked](#), [LinkFields](#), [Link](#), [LinkIndicator](#), [LinkShadow](#)

Launching applications

[LaunchCommand](#), [LaunchStartDir](#), [LaunchFlags](#), [IsLaunched](#), [Launch](#), [LaunchIndicator](#), [LaunchShadow](#)

Printing charts

[PrintOut](#), [PrintSelected](#), [Printer](#), [PrintBlankPages](#)

Redrawing a chart

[NoRepaint](#), [Repaint](#)

Adjusting the page layout

[Height](#), [MarginBottom](#), [MarginLeft](#), [MarginRight](#), [MarginTop](#), [Orientation](#), [PageHeight](#), [PageLayout](#), [PageOrder](#), [PageWidth](#), [PaperSize](#), [PrintBlankPages](#), [Width](#)

Displaying Master Items

[UpdateDateAndTime](#), [ChartName](#), [Date](#), [DateStyle](#), [Logo](#), [LogoPathname](#), [MasterItems](#), [PageNumber](#), [Range](#), [Text1](#), [Text2](#), [Time](#), [PageNumberShown](#), [Text1Shown](#), [Text2Shown](#), [TimeShown](#), [LogoShown](#), [ChartNameShown](#), [DateShown](#), [HideAll](#), [ShowAll](#)

Viewing a chart

[PageCount](#), [ScrollLeft](#), [ScrollTop](#), [ScrollPage](#), [ScrollPosition](#), [ZoomPercentage](#), [FullScreen](#), [CancelFullScreen](#)

Using guidelines

[GuidelinesOn](#), [AddHorizontalGuideline](#), [AddVerticalGuideline](#), [ClearGuidelines](#)

Sizing a chart window

[Maximize](#), [Minimize](#), [Restore](#), [View](#)

Changing the view magnification

[View](#), [ZoomPercentage](#)

Setting units of measure

[Units](#)

Doing a presentation

[FullScreen](#), [CancelFullScreen](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Working with Objects

Related

### Task Properties/Methods

Getting an object in a chart

[Count](#), [Item](#), [ItemFromAll](#), [ItemFromAttachments](#), [ItemFromLines](#), [ItemFromFieldValue](#),  
[ItemFromNumber](#), [ItemFromShapes](#), [ItemFromSelection](#), [ItemFromText](#), [ItemFromUniqueID](#),  
[ResetSearch](#), [Valid](#)

Selecting objects in a chart

[Selected](#), [Select](#), [SelectShapeType](#)

Deselecting objects in a chart

[DeselectAll](#)

Finding the number of selected objects

[SelectedLineCount](#), [SelectedObjectCount](#), [SelectedOtherCount](#), [SelectedShapeCount](#)

Identifying an object

[Type](#), [UniqueID](#), [ShapeName](#)

Cutting, copying, and pasting objects

[Copy](#), [Cut](#), [Paste](#), [PasteLink](#), [Duplicate](#), [Clear\\_](#), [PasteSpecial](#), [ClipboardFormatAvailable](#)

Executing an embedded or linked object

[OLE](#), [ObjectType](#), [DoVerb](#)

Inserting objects from a file

[InsertObjectFromFile](#)

Moving objects

[Bottom](#), [CenterX](#), [CenterY](#), [Left](#), [Top](#), [Right](#)

Resizing objects

[Height](#), [StretchType](#), [Width](#)

Restoring a bitmap

[RestorePicture](#)

Changing the display order of objects

[ToBack](#), [ToFront](#)

Setting the current drawing position

[DrawDirection](#), [DrawPositionX](#), [DrawPositionY](#)

Redrawing an object

[Repaint](#)

Undoing a change

[Undo](#), [UndoAvailable](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Working with Shapes

Related

### Task Properties/Methods

Using the shape palette

[CurrentShapePalette](#), [ShapePaletteVisible](#), [CurrentShape](#)

Drawing shapes

[DrawShape](#), [DrawDirection](#), [DrawPositionX](#), [DrawPositionY](#), [DrawSpacingX](#), [DrawSpacingY](#),  
[ShapeName](#), [CurrentShape](#), [SetDefaults](#)

Connecting shapes with lines

[DrawLine](#), [Source](#), [Destination](#), [ReconnectSource](#), [ReconnectDest](#), [CurrentLineRouting](#)

Moving shapes

[Bottom](#), [Left](#), [Right](#), [Top](#)

Selecting shapes

[SelectedShapeCount](#), [DeselectAll](#), [ItemFromShapes](#), [Selected](#), [SelectShapeType](#)

Formatting shapes

[Color](#), [BorderColor](#), [BorderStyle](#), [BorderWidth](#), [FillColor](#), [FillPattern](#), [ShadowColor](#), [ShadowOffset](#),  
[ShadowStyle](#), [Shape](#), [SetDefaults](#)

Numbering shapes

[NextNumber](#), [NumberFont](#), [Number](#), [NumberShown](#), [Rename](#)

Replacing shapes

[ReplaceShape](#)

Adding notes to shape

[NoteIndicator](#), [NoteShadow](#), [NoteFont](#), [NoteText](#), [NoteTextLF](#)

Adding text to shapes

[TextAlignment](#), [FitShapeToText](#), [Text](#), [TextLF](#), [Font](#)

Resizing shapes

[Height](#), [Width](#), [StretchType](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Working with Lines

Related

### Task Properties/Methods

Drawing lines

[DrawFreeLine](#), [DrawLineToOneObject](#), [DrawLine](#)

Connecting existing lines to shape

[Source](#), [Destination](#), [ReconnectSource](#), [ReconnectDest](#), [Line](#)

Setting line routing

[CurrentLineRouting](#), [Type](#)

Formatting lines

[Color](#), [SourceArrowColor](#), [StemColor](#), [DestArrowColor](#), [SourceArrowSize](#), [StemWidth](#),  
[DestArrowSize](#), [StemStyle](#), [SourceArrowStyle](#), [DestArrowStyle](#), [Line](#), [LineCrossoverStyle](#),  
[LineCrossoverSize](#), [SetDefaults](#)

Displaying nodes on connecting lines

[ShowNodesOnLines](#), [LineCrossoverStyle](#), [LineCrossoverSize](#)

Attaching text to lines

[Line](#), [AttachText](#), [AttachedToLine](#), [UnattachFromLine](#), [TextBlock](#)

Deleting lines

[DeleteLines](#), [Clear](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Working with Text Related

### Task Properties/Methods

Creating text blocks

[DrawTextBlock](#), [DrawPositionX](#), [DrawPositionY](#)

Adding text to a shape

[Text](#), [TextLF](#), [FitShapeToText](#)

Adding notes to a shape

[NoteText](#), [NoteTextLF](#), [NoteFont](#), [NoteIndicator](#), [NoteShadow](#)

Adding text to a line

[DrawTextBlock](#), [AttachText](#), [AttachedToLine](#), [UnattachFromLine](#)

Formatting text

[Size](#), [Name](#), [Bold](#), [Italic](#), [Underline](#), [Strikethrough](#), [Color](#), [Opaque](#), [TextAlignment](#), [SetDefaults](#)

Spell checking

[Spelling](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Working with Data Fields

Related

### Task Properties/Methods

Adding data fields to a chart

[Add](#), [Name](#), [Format](#), [AccumulationMethod](#), [Hidden](#), [Type](#), [FieldTemplates](#)

Deleting data fields from a chart

[DeleteField](#)

Setting data field preference

[FieldNamesHidden](#), [FieldPlacement](#), [FieldsOpaque](#), [FieldsHoursPerDay](#), [FieldsDaysPerWeek](#), [FieldFont](#)

Working with data field values

[Value](#), [Item](#), [Accumulation](#), [ItemFromFieldValue](#), [Empty](#), [FieldViewerVisible](#), [FieldValues](#), [Count](#), [IsEmpty](#), [FormattedValue](#)

Viewing the legend

[ShowLegend](#), [Accumulation](#)

Using linked field data

[LinkFields](#), [UpdateFields](#), [LinkIndicator](#), [LinkShadow](#), [IsLinked](#), [LinkedChartName](#), [Link](#)

Getting a data field

[Count](#), [Item](#), [FieldTemplates](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Using Color

Related

### Task Properties/Methods

Color representation

[BasicColor](#), [MakeRGB](#)

Setting shape colors

[FillColor](#), [Color](#), [BorderColor](#), [ShadowColor](#), [SetDefaults](#)

Setting line colors

[Color](#), [SourceArrowColor](#), [DestArrowColor](#), [StemColor](#), [SetDefaults](#)

Setting text color

[Color](#), [SetDefaults](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Customizing Menus

Related

### Task Properties/Methods

Adding a new menu

[AddMenu](#)

Getting a menu item

[Count](#), [Item](#)

Adding a menu item

[AppendItem](#)

Deleting a menu item

[DeleteItem](#), [DeleteAll](#), [Visible](#)

Hiding a menu item

[Visible](#)

Changing menu text

[Text](#)

Disabling a menu item

[Enabled](#)

Checking a menu item

[Checked](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

## Tasks: Special Programming Features

Related

### Task Properties/Methods

Displaying a wait hourglass

[Hourglass](#)

Displaying a percent gauge

[PercentGaugeValue](#), [PercentGauge](#), [PercentGaugeCancelled](#), [HidePercentGauge](#)

Displaying a message

[MsgBox](#)

Using ABC Events

[RegisterEvent](#), [UnRegisterEvent](#), [ChartTypeShutdown](#)

Sending mail

[SendMail](#)

## Related Topics

### Description

[Properties and Methods by Task](#)

