Lead Icon in a Composite

The first icon in any composite is called the "lead" icon. It marks the beginning of the composite.

A composite icon is one icon in the library that actually represents a mini-structure of several icons. Composites provide a way for saving groups of icons that represent a portion of a structure that is commonly used in many applications or repeatedly used in one. IconAuthor contains some composite icons in the initial library. You can also create an save your own composites.

Related Topics:

CD-Audio Icon
MIDI Icon
ObjMenu Icon
WaveAudio Icon

Beep Icon

The Beep icon generates a brief computer sound. To generate a number of beeps, use multiple Beep icons in sequence (or create a Beep icon in a Loop)

Suggested Uses:

- Use a low tone to provide negative feedback, to signal the user that time is running out or to indicate that the system is waiting for input.
- Use a high tone to indicate that a menu selection has been made or to provide positive feedback.

Beep Type

Specify the type of beep you want to generate: **high**, **low**, or a variable.

Drop-down List Box Items:

high - Creates a high tone.

low - Creates a low tone.

Variable Selector

Box Icon

The Box icon dynamically draws a box.

You can control whether the box is filled and outlined, or outlined only. By default, the outline color is black and the fill color is white. To create a box using alternative colors, precede the Box icon with a Color icon. The fill and outline colors you specify are in effect until you use another Color icon. (If you use the Color icon to specify a fill color of TRANSPARENT, the underlying screen display will show through the interior of the box.)

Suggested Uses:

- Create a box that is a border around the screen or around a part of the screen.
- Mask a box-shaped area of the screen.
- Create a square bullet to emphasize an item on the screen.

Content Editor Text Boxes:

Upper Left Corner
Lower Right Offset
Line Width
Filled or Outline

Upper Left Corner

Specify the location of the upper left corner of the box.

Acceptable values are: **upper left** or 2 numbers, separated by commas, that define a point on the screen. The numbers are the x,y coordinates of the upper left corner of the area. You can use variables for any of the coordinates.

Drop-down List Box Items:

upper left - Defines the upper left corner of the box as the upper left corner of the screen.

Area Editor

Variable Selector

Lower Right Offset

Specify the width and height (in pixels) of the box. Acceptable values are: **lower right** or two numbers, separated by commas, that represent the width and height respectively; or a variable.

Note: A value automatically appears in this text box if you use the Area Editor to specify the upper left corner of the box.

Drop-down List Box Items:

lower right - Defines the width and height of the box so that its lower right corner is located at the lower right corner of the screen.

Variable Selector

Line Width

Specify the width (in pixels) of the border of the box. If you are drawing an outlined box, use a line width of 1 pixel or more. If you are drawing a filled box, use a line width of 0 pixels or more. (If the line width of a filled box is equal to 0, it has no border and uses only a fill color.)

Drop-down List Box Items:

assorted whole numbers

Variable Selector

Filled or Outline

Indicate whether the box is filled and outlined, or outlined only. Acceptable Values are **filled**, **outline**, or a variable.

Drop-down List Box Items:

filled - Creates a filled box. If the line width of a filled box is greater than 0, it uses a fill color and an outline color. If the line width of a filled box is equal to 0, it has no border and uses only a fill color.

outline - Creates an outlined box. The width must be greater than 0.

Variable Selector

Branches Icon

The Branches icon is a composite icon that causes the application to take a particular "branch" or path of execution when a user selects one or more options.

By default, the Branches icon consists of a lead icon called "Branches" and four If icons, labeled "1" through "4". Each of these If icons marks the start of a branch that can potentially be taken. (You can edit the structure so that there is a smaller or larger number of branches.)

When IconAuthor finds the Branches icon it goes to the first If icon, labeled "1" and checks to see if a condition is true. If the condition is true, execution flows downward from that icon. If the condition is false, execution flows to the right and IconAuthor checks to see if the condition in the second If icon, labeled "2" is true. The process continues until one of the branches is executed, or until all of the Branch conditions have been tested.

If one of the Branch conditions is true, that branch executes. When it finishes, execution flows to the icon just below the composite Branches icon. If none of the Branch conditions is true, no branch is executed, and execution flows to the icon just below the composite Branches icon.

Suggested Uses:

- Display different information depending upon the selection a user makes from a menu.
- Display a particular message or test score based on user performance of a particular task.
- Let the user choose a subject from a menu to learn more about it.

Content Editor Text Boxes:

Condition 1
Test
Condition 2
Condition Type

Related Topics:

Comparing Numbers
Comparing Non-Displayable Characters
Character String Matching

Condition 1

Specify the name of a variable that you want to compare to the value in the Condition 2 text box.

Drop-down List Box Item:

Variable Selector

Test

Specify the test that you want to use to compare the value in the Condition 1 text box and the value in the Condition 2 text box. Acceptable Values are: EQ, NE, LT, or GT

Drop-down List Box Items:

EQ - Tests if Condition 1 is *equal to* Condition 2.

NE - Tests if Condition 1 is *not equal to* Condition 2.

LT - Tests if Condition 1 is *less than* Condition 2.

GT - Tests if Condition 1 is *greater than* Condition 2.

Condition 2

Specify the value to which you want to compare the variable in Condition 1. Acceptable Values are a numeric value (including negative and decimal numbers), an alphanumeric character, an alphanumeric string, a non-displayable key (such as the RETURN key), or the coordinates of a rectangular area.

Drop-down List Box Items:

- Area Editor
- Object Name Selector
- Object Event Selector
- Variable Selector

Condition Type

Specify the kind of data being compared. Acceptable Values are: alphabet, numeral, rectangle, or a variable.

Drop-down List Box Items:

alphabet - Indicates that the values you are comparing are alphanumeric.

numeral - Indicates that the values you are comparing are numeric.

rectangle - Indicates that the values you are comparing are rectangle coordinates.

Variable Selector

Comparing Numbers

When you compare numbers, if the Condition Type is numeral then the number 125 is less than 1150; if the Condition Type is alphabet then the number 125 is greater than (alphabetically after) 1150.

Related Topics:

Branches Icon
If Icon

Comparing Non-Displayable Characters

The If icon can test for non-displayable character input to execute a different branch depending upon the key that the user pressed.

As an example, include an Input icon in your application that will store the string representing the key the user presses in a variable called @INPUT. Then use a composite Branches icon to take a particular path depending on which key is pressed. The first If icon might test whether the value in @INPUT is equal to "return", the second If icon can test whether the value in @INPUT is equal to "ins" for insert, and so on.

Specify alphabet in the Condition Type text box to test for a non-displayable character. The non-displayable characters and the strings that they generate are as follows:

Key Name	String	Key Name	String
Backspace	bs	Pause	pause
Cancel	cancel	Return	return
Clear	clear	Tab	tab
Delete	del	Page Down	next
End	enter	Page Up	prior
Enter	enter	left arrow	left
Esc*	esc	right arrow	right
Execute	execute	up arrow	up
Help	help	down arrow	down
Home	home	function keys	f1, f2, ..., f12
Insert	ins		

* The Esc key is used in the IconAuthor Authoring system to escape from a running application and return to the IconAuthor window. You can, however, still use the Esc key in your applications. In the IconAuthor Presentation system, the Esc key functions just like any other non-displayable character.

If Num Lock is on, inputs from the numeric keypad generate the following strings:

Key Name	String	Key Name	String
0	numpad0	5	numpad5
1	numpad1	6	numpad6
2	numpad2	7	numpad7
3	numpad3	8	numpad8
4	numpad4	9	numpad9

Note: The spacebar generates a space character, not the word "space".

The following keys *do not* generate an input character:

CapsLock	Shift
CTRL	Alt
Print Screen	Scroll Lock
Pause	Num Lock

Related Topics:

Branches Icon
If Icon

Character String Matching

An If icon can test whether a string the user enters (assigned to Condition 1) matches the value in Condition 2. You can control the precision used in string matching. For example, you can require the user to enter a string that 1) matches the Condition 2 string character for character and/or 2) matches the exact use of upper and lower case in Condition 2.

You can also control string matching so that the user can enter a value that is similar to the string in Condition 2, and is interpreted as a match. For example, if the answer to a test question is the string "blue", you can control string matching so that the string "light blue" is interpreted as a correct answer.

Switches are used in the value you assign to Condition 2 to control string matching.

The following switches are available:

/$ This switch controls whether or not the string the user enters is tested for exact matching of upper and lowercase.

Example: If Condition 2 is /$ABc, the string the user enters must match the use of case exactly in order for the condition to be true.

/# This switch follows /$. /# resets the matching control so that another part of the string in Condition 1 does not have to match the exact use of case in Condition 2.

Example: If Condition 2 is /$F/#red, the first switch indicates that the following "F" must be uppercase. The second switch indicates that the "red" can be either upper or lowercase.

/? This switch represents a single character wild card.

Example: If Condition 2 is F/?ed, the character between the "F" and the "e" can be any upper or lower case character.

/* This switch represents multiple wild card characters.

Example: If Condition 2 is F/*d, any number of characters can appear between the "F" and the "d". (The user can enter "Fred" or "Friend" and a match will still occur.)

/% This switch searches for the group of characters that follow the /%.

Example: If Condition 2 is /%Fred, any string will be a match as long as it contains "Fred". (The user can enter "Fred", "Mr. Fred Smith", or "My name is Fred".)

Related Topics:

Branches Icon
If Icon

CD-Audio Icon

The CD-Audio icon is a composite that allows you to play CD-Audio if you are using the Multimedia Extensions software and a CD-ROM drive.

Hint: An even quicker and easier way to play audio is via the SmartObject Editors Audio object.

Five MCI icons form the backbone of the CD-Audio composite. Although you must be familiar with the MCI syntax in order to fully take advantage of the MCI feature, the CD-Audio composite already contains some values so that you can quickly start playing CD-Audio as part of your IconAuthor applications. Once you become familiar with the MCI syntax you can customize and vary the commands. For information on the MCI command syntax open the help file called MCISTRWH.HLP.

The composite contains a mini-structure of icons:

1. MCI icon: Contains the command **open cdaudio alias cd**, where: **open cdaudio** initializes the device and **alias cd** specifies the name "cd" as an alternate name for the cdaudio device type.
2. MCI icon: Contains the command **set cd time format tmsf** which sets the time format for play to tracks, minutes, seconds, and frames.
3. MCI icon: Contains the command **play cd from 1 to 1:00:30** which starts the CD playing track 1 for 30 seconds.
4. Input icon: Causes execution flow to stop at this point and wait for the user to provide input. This icon specifies that the entire screen is input selectable. That means that the user can click anywhere or press any key to cause execution flow to continue.
5. MCI icon: When the user clicks, execution flows to this fourth MCI icon which contains the command **pause cd** to pause the CD from playing.
6. MCI icon: Contains the command **close cd** to suspend playback and relinquish access to the device.

Hint: If you want to use this composite to play a CD while some other activity is occurring, replace the Input icon with one or more alternative icons. For example, if you use a Display icon (in place of the Input icon) to run an animation script, the audio will play, the animation will run, and when the animation completes, the audio will stop.

Content Editor Text Box:

The lead icon in the CD-Audio composite is labeled "CD-Audio" and contains only one text box "Composite Name". Enter a different name in this text box to customize the name of the composite.

Related Topics:

MCI Icon
Input Icon

Circle Icon

The Circle icon dynamically draws a circle.

You can control whether the circle is filled and outlined, or outlined only. By default, the outline color is black and the fill color is white. To create a circle using alternative colors, precede the Circle icon with a Color icon. The fill and outline colors you specify are in effect until you use another Color icon. (If you use the Color icon to specify a fill color of TRANSPARENT, the underlying screen display will show through the interior of the circle.)

Suggested Uses:

- Mask a circular area of the screen.
- Create a bullet to emphasize an item on the screen.

Content Editor Text Boxes:

Center Location
Radius
Line Width
Filled or Outline

Center Location

Specify the location of the center of the circle. Acceptable Values are: **upper left**, a pair of screen coordinates, separated by a comma; or a variable.

Drop-down List Box Items:

upper left - defines the center of the circle as the upper left corner of the screen.

Location Editor

Variable Selector

Radius

Specify the radius (in pixels) of the circle. Acceptable Values are: any whole number (including 0) or a variable.

Drop-down List Box Items:

assorted whole numbers - Frequently used line widths.

Variable Selector

Line Width

Specify the width (in pixels) of the border of the circle. If you are drawing an outlined circle, use a line width of 1 pixel or more. If you are drawing a filled circle, use a line width of 0 pixels or more. (If the line width of a filled circle is equal to 0, it has no border and uses only a fill color.) Acceptable Values are any whole number (including 0) or a variable.

Drop-down List Box Items:

assorted whole numbers - Frequently used line widths.

Variable Selector

Filled or Outline

Indicate whether the circle is filled and outlined, or outlined only. Acceptable Values are: **filled**, **outline**, or a variable.

Drop-down List Box Items:

filled - Creates a filled circle. If the line width of a filled circle is greater than 0, it uses a fill color and an outline color. If the line width of a filled circle is equal to 0, it has no border and uses only a fill color.

outline - Creates an outlined circle. The value entered must be greater than 0.

Variable Selector

Clear Icon

The Clear icon clears the screen to a specified color.

Suggested Uses:

- Clear the screen between SmartObject displays. By default, a SmartObject file has a transparent background, causing the previous text to appear behind the current text unless you clear the screen between displays. (Hint: To clear just a small part of the screen, mask that area with a dynamically generated shape.)
- Clear the screen to the "transparent" color before you display video. Video is visible on the screen wherever the transparent color occurs. The particular color that is considered transparent is determined by the video overlay board installed on your system. Refer to the documentation for your overlay board to determine which color is transparent. (Frequently the transparent color is black.)
- Clear the screen to a color between graphic displays for a different transition effect.

Important: If you display live objects (from a SmartObject file) the Clear icon will not remove these objects from view. You must use make the objects invisible (using an ObjSet icon) or you must delete them (using an ObjDelete icon).

Content Editor Text Box:

Clear Screen To

Clear Screen To

Specify the color to which you want to clear the screen. Acceptable Values are a color name, an RGB (Red Green Blue) value, or a variable.

Drop-down List Box Items:

assorted colors - Frequently used colors such as black, red, etc.

Color dialog box

Variable Selector

Color Icon

The Color icon sets the outline and fill color for dynamic graphics (the Box, Circle, Ellipse, and Line icons) and sets the color for text displays (the Text and Write icons).

By default, the outline color is black and the fill color is white. To create a dynamic shape or text using alternative colors, precede the icon with a Color icon. The fill and outline colors you specify are in effect until you use another Color icon.

Suggested Uses:

- Use different colors to fill dynamic shapes, such as circles and rectangles.
- Use different colors when you create dynamic text (the Write icon) or when you generate text from an ASCII file (the Text icon).
- Create text with a transparent fill color so that the text has color, but the background of the text has no color. The text appears to lie directly on top of the current display.

Content Editor Text Boxes:

Outline Color
Fill Color

Outline Color

Specify the color you want to use to display text or lines, and to create the border of circles, rectangles, and ellipses. Acceptable values are a color name, an RGB (Red Green Blue) value, or a variable.

Drop-down List Box Items:

assorted colors - Frequently used colors such as black, red, etc.

Color dialog box

Variable Selector

Fill Color

Specify the color you want to use to fill dynamically generated circles, rectangles, and ellipses, and to create the background color for text. Acceptable values are: a color name, an RGB (Red Green Blue) value, TRANSPARENT, or a variable.

Drop-down List Box Items:

assorted colors - Frequently used colors such as black, red, etc.

TRANSPARENT - Creates *no* fill color. Causes the underlying screen display to show through the interior of a dynamically generated shape, or behind text displayed with the Write icon or the Text icon.

Color dialog box

Variable Selector

DDE Icon

The DDE (Dynamic Data Exchange) icon allows IconAuthor to communicate with other Microsoft Windows programs. As an example, IconAuthor can use a series of DDE icons to communicate with a Microsoft EXCEL spreadsheet, inserting values into data cells or to retrieving values from data cells.

Suggested Uses:

+ Provide data to another program.
+ Retrieve data from another program.
+ Execute commands in another program.

DDE is a communication feature of Microsoft Windows. When two applications use Microsoft's DDE mechanism to communicate one is called the client and the other is called the server. The client is the program that initiates communication. The server is the program that is being called. DDE can be used to cause the server to open a file, request data from the server, provide data to the server, or cause the server to close a file.

IconAuthor is DDE-aware and the DDE icon allows it to function as a client. (IconAuthor and Present can also function as a server.) Each conversation (communication between IconAuthor and the other application) is initiated by IconAuthor and is assigned to a unique channel. Up to 16 channels can be open at any one time. If your IconAuthor application requires more than 16 channels, you must terminate one of the 16 in order to open another.

In order to create a conversation between IconAuthor and another application, the other application must be running.. (The other program can be running using the Program icon.)

The first DDE icon initiates the conversation, indicating the application with which you want IconAuthor to communicate. You specify an application and a topic**.** Valid application and topic values are determined by the server application. For example, EXCEL recognizes the text value **EXCEL** as its application parameter and it recognizes the text value **SYSTEM** or the name of any currently open spreadsheet (for example, SALES.XLS) as its topic parameter.

Subsequent DDE icons can be used to send or receive information. In those cases, you indicate which data you are manipulating by specifying a parameter called item. Like the application and topic parameters, valid item identifiers are determined by the server application. As an example, a conversation could include a DDE icon that requests a value from an EXCEL spreadsheet data cell. You specify **REQUEST** in the Command field, and **R1C2** (as the item) in the Parameters field to indicate that you want the server to return the value in the data cell at row 1, column 2. The value in this cell is returned to the variable you specify in the Result Variable field.

All data is returned from the server as a text string. A command returns an error if it is unsuccessful. Errors appear in the form ***DDE ERROR x**, where x is an error number. If no data is returned by the command, and no error occurred, the string ***DDE OK** is returned.

When communication is complete, a DDE icon terminates the conversation or conversations that occurred.

Content Editor Text Boxes:

Command
Channel
Parameters
Time Limit
Result Variable

Command

Specify the DDE command you want to give. Acceptable values are: INITIATE, REQUEST, POKE, TERMINATE, or a variable

Drop-down List Box Items:

INITIATE - Opens a conversation channel. Indicate the application and topic in the Parameters field. The INITIATE command returns the channel identifier to the variable you specify in the Result Variable field. Use this variable in future commands that will be part of this conversation. (You do not specify a value for the Channel field when you give an INITIATE command, because a Channel value has not yet been assigned.)

Note: The application with which you are initiating a conversation must have already been executed. The INITIATE command does not execute the application.

EXECUTE - Sends a command (that you specify in the Parameters field) to the server application. For example, you can use this command to load a spreadsheet after you have initiated a conversation with EXCEL. When using this command, use the Channel field to specify the variable that contains the channel identifier for this conversation that was assigned at initiation. Use the Parameters field to specify the command to send to the server, for example, [open("e:\excel\sales.xls")]**.** Note that this command must be surrounded by brackets.

POKE - Sends data to the server application. When you use this command, use the Channel field to specify the variable that contains the channel identifier for this conversation that was assigned at initiation. Use the Parameters field to indicate the data item identifier for the data that you want to send to the server. Acceptable values for items are defined by the server. There is no data value returned for the POKE command. You can however view the string that is returned, such as ***DDE OK by including a variable in the Result Variable field. After you run the IconAuthor application, check the value stored in this variable. To do this, choose Window Contents from the View menu, and choose User Variables.

REQUEST - requests data from the server. When you specify this command, use the Channel field to specify the variable that contains the channel identifier for this conversation that was assigned at initiation. Use the Parameters field to indicate the data item identifier for the data being requested. Acceptable values for items are defined by the server. The data requested is returned to the variable you specify in the Result Variable field.

The data you request can be a single value, or a range of values.

TERMINATE - ends the conversation specified in the Channel field. If 16 conversations are already being conducted, you must terminate one conversation in order to initiate another.

Variable Selector

Channel

The channel determines the conversation to which the command is sent. It is empty for the INITIATE command. Specify the variable that contains the channel identifier of the conversation. The channel identifier is assigned to the variable in the Return Variable field after the INITIATE command.

Drop-down List Box Item:

Variable Selector

Parameters

List the parameters required by the command specified in the Command field. If a command requires multiple parameters, separate the parameters by a comma. If a command does not require multiple parameters, you can use commas as part of a single parameter and they are properly interpreted.

Drop-down List Box Item:

Variable Selector

Time Limit

Use the Time Limit field to specify the number of seconds IconAuthor should wait for an acknowledgement from the server. The time limit must be less than or equal to 60 seconds.

Drop-down List Box Item:

Variable Selector

Result Variable

Use the Result Variable field to specify a variable that receives the resulting value of the DDE command given to the server.

All data is returned in the form of a text string. If the communication is successful and if no data is returned be the command, the string ***DDE OK** is returned. The command returns an error if it is unsuccessful. Errors take the form ***DDE ERROR x**, where x is an error number.

Drop-down List Box Item:

Variable Selector

The following table describes possible error messages:

Error Number	Description
22055	Bad channel variable
22056	Bad command field
22057	Null field
22058	Time out
22059	Bad DDE format
22060	Poke failed
22061	Add atom failure
22062	Global lock failed
22063	Global allocation failed
22064	Request failed
22065	No return variable
22066	Unexpected DDE message

Present as a DDE Server

DDE (Dynamic Data Exchange), a communication feature of Microsoft Windows, allows your IconAuthor applications to communicate with other DDE-enabled Microsoft Windows programs. For example, your applications can communicate with a Microsoft EXCEL spreadsheet or a Word for Windows document.

When two programs use Microsoft's DDE mechanism to communicate one is called the **client** and the other is called the **server**. The client is the program that initiates communication. The server is the program that is being called. DDE can be used to cause the server to open a file, request data from the server, provide data to the server, or cause the server to close a file. An IconAuthor application can act as a DDE client or server.

IconAuthor/Present as DDE Client

In order to use IconAuthor/Present as a client, use DDE icons in your structure.

Present as DDE Server

Present can act as a DDE server. If Present is running, another DDE-enabled program (acting as the client) can:

Launch an IconAuthor application.
Set an IconAuthor variable.
Get the contents of an IconAuthor variable.
Close an IconAuthor application.

As an example, you can create an IconAuthor application that provides interactive online Help for a user working with an EXCEL spreadsheet. With Present running in the background, when the user needs information, he or she performs an action that causes the spreadsheet application to initiate communication with Present. Once communication starts, the spreadsheet can execute the specific IconAuthor application. The application can display different kinds of information (such as graphics, text, and animation). During the course of execution, the spreadsheet can set an IconAuthor variable so that a

particular course of action is taken in the IconAuthor application. Or, the spreadsheet can retrieve a value from the application, that will be used in the spreadsheet.

Conversations with Present as the Server

The DDE communication between two programs is called a **conversation**. In order to create a conversation, you must be familiar with 1) how the client application uses DDE, for example, how do you initiate a conversation from within an EXCEL spreadsheet, and 2) which application and topic parameters are supported by Present. To learn about the DDE interface of the client application, see the documentation that accompanies that program.

The following paragraphs describe the commands and parameters (applications and topics) recognized by Present.

DDE Initialize: This command allows a client application to establish a DDE communication connection with Present. To establish a connection, the user must supply the application name *IAUTHOR* and a conversation topic of NULL or *System.*

DDE Request: This command allows a client application to extract data from your IconAuthor application. The content of the data depends upon the DDE Item requested. The following DDE Items are supported by Present.

SysItems: This Item returns a list of System-topic items supported by Present. Each item is delimited by a TAB (hex 09) character. The last Item in the list is NULL terminated.

ReturnMessage: This Item returns a DDEML error value for the last DDE transaction. This numerical string provides the client application with a mechanism for determining DDE transaction failures. A value other than zero indicates the failure error number.

Status: This Item returns the current status of Present. If a string of "Busy" is returned, Present is currently executing an application. A return string of "Ready" indicates that Present is idle and waiting for an application file to run.

Formats: This Item returns a list of clipboard format values Present can process. Currently, only CF_TEXT is supported.

IconAuthor Variable: This item returns the contents of an IconAuthor variable. The DDE client must supply the IconAuthor variable name as the DDE Request Item.

DDE Poke: This command allows the client application to set unsolicited data within your IconAuthor application. The client supplies an IconAuthor variable name as the Poke Item and the variable contents as the Poke Data.

DDE Execute: This command allows a client application to send a command for Present to process. The following two

commands are currently supported.

[load("*filename*")] This command allows the client application to load and run an IconAuthor application in Present. The *filename* must be a full DOS path to the application file. If an application file is currently running, a DDE busy response will be returned.

[stop] This command allows the client application to stop a running application file. If the application running is waiting for some form of user input, the running application will be terminated after the input has been acquired or when a time-out has occurred.

DDE Terminate Terminates the current client connection with the Present DDE server.

DllCall Icon

A Dynamic Link Library (DLL) is a special executable file that consists of a "library" of one or more functions. These functions can be called by another executable file. Through two icons, DllLink and DllCall, your IconAuthor applications can communicate with functions in DLLs.

To understand the purpose of the DllCall icon consider how it works in conjunction with the DllLink icon. You use the DllLink icon to load a prototype file and (in some cases to pre-load one or more DLLs). You then use a DllCall icon to call a function that is described in the prototype file. You cannot call a function unless the prototype file that describes that function has been loaded.

Suggested Uses:

Use a DLL function to display a custom dialog box.

Use a DLL function to support a device not currently supported by IconAuthor.

Prerequisites for Using DLL Icons:

Because working with DLLs requires a sophisticated understanding of programming you must have the assistance of an experienced programmer in order to use the DLL icons. Only a programmer has the ability to locate or create the appropriate DLL to suit the needs of your application.

Content Editor Text Boxes:

Return Variable
DLL Function

Return Variable

Specify the name of the variable that will receive the value being returned from the called function. If the function does not return a value leave this text box empty.

Drop-down List Box Item:

Variable Selector

DLL Function

Specify the function you are calling and the values you are sending to it. For example:

MessageBox(Null,@MESSAGE,"Demo",3);

This is the function MessageBox which requires four values. The values being sent must be listed in the correct order within the parentheses and multiple values must be separated from one another by a comma. If the function does not expect any values state the function name followed by empty parentheses. End the line with a semi-colon (;).

When you choose OK to close the DllCall icon Content Editor a message will be displayed if there is a discrepancy between the function you specify and the function description in the prototype file. You have the opportunity to go back and correct the Content Editor information or accept the Content Editor values and edit the prototype file.

Note: If the programmer has set up an alias for the function name in the prototype file, optionally, use the alias.

DllLink Icon

The main purpose of the DllLink icon is to load function prototypes so that IconAuthor can know how to call functions within DLLs. Also, you can use DllLink icons to pre-load DLLs into memory. Typically, you use the DllLink icon to load a prototype file and then use a DllCall icon to call a function that is described in that file.

The DllLink icon uses two IconAuthor system variables: @_ERROR and @_ERROR_STRING. When the DllLink icon executes successfully @_ERROR contains the value 0. When the DllLink icon executes and an error occurs @_ERROR contains a non-zero value. Also, @_ERROR_STRING is assigned a message that describes the nature of an error that has occurred. (While you are authoring, you can view the contents of @_ERROR and @_ERROR_STRING by choosing Window Contents from the View menu and then choosing System Variables.)

Suggested Uses:

Use a DLL function to display a custom dialog box.

Use a DLL function to support a device not currently supported by IconAuthor.

Prerequisites for Using DLL Icons

Because working with DLLs requires a sophisticated understanding of programming you must have the assistance of an experienced programmer in order to use the DLL icons. Only a programmer has the ability to locate or create the appropriate DLL to suit the needs of your application.

Content Editor Text Boxes:

Prototype Filename
Prototype Action
DLL Names
DLL Action

Prototype Filename

Specify the name of the prototype file that identifies and defines the DLL functions that your application is going to use.

Drop-down List Box Items:

Directory
Variable Selector

Prototype Action

The only action available is to load a prototype file. If you do not want to load a prototype file leave this text box empty.

Drop-down List Box Items:

LoadPrototypes - Loads (declares) function prototype to make them available for calling.
Variable Selector

DLL Names

Optionally, enter the name of the DLL you want to load or free (as specified in the DLL Action text box). In most situations you do not need to use this text box.

Drop-down List Box Items:

All - Indicates that all DLLs listed in the prototype file (specified in the Prototype Filename text box) are to be loaded or freed (as specified the DLL Action text box). If you specify All and do not enter a filename in the Prototype Filename text box, all DLLs in previously loaded prototype files are loaded or freed.

Variable Selector

DLL Action

Specify the action to be taken on the file(s) you specified in the DLL Names text box. You can either load DLLs or free them from memory. As expressed under "DLL Name," you typically do not need to use this text box.

Drop-down List Box Items:

LoadLibrary - Loads one or more DLLs into memory.

FreeLibrary - Frees one or more DLL's from memory.

Variable Selector

Database Icon

The Database icon lets your application create and work with dBASE database files. Each Database icon you include in your application can issue a different database command.

Important: As an alternative, the SmartObject Editors Database object allows for interaction with a greater variety of database file formats. It also provides an optional user interface for browsing through data.

Content Editor Text Box:

Database Commands

Related Topic:

Steps for Working with the Database Icons

Steps for Working the Database Icons

There are three steps to perform in order to work with an IconAuthor database:

1. Planning the database.

   Sometimes your IconAuthor application uses a database that already exists. Perhaps the database was previously created either with IconAuthor, or with dBASE III Plus or dBASE IV. Although you don't have to plan a database that already exists, you have to determine whether the information it contains meets the needs of your application.

2. Creating the format file that organizes the database.

   If your IconAuthor application uses a database that was created using dBASE II Plus or dBASE IV, you do not need to create a format file.

3. Using Database icons in your application to create the database if necessary, and then access and manipulate the database.

Database Commands

Use the Database icon to specify a command you want to use to affect the database. You can choose a command from the drop-down list box or you can use a variable that contains a command. The syntax of most commands requires you to include additional parameters.

Each available database command is available in the drop-down list box:

- add command
- close command
- create command
- delete command
- exist command
- locate command
- next command
- replace command
- store command
- use command
- Variable Selector

exist command

Syntax: **exist *database_filename***

To begin using a database file, use the exist command to test whether the database file exists. The database file may already exist if it was created:

- Using dBASE III Plus or dBASE IV.
- Using IconAuthor when the same application was run previously.
- Using IconAuthor when a different application (that uses the same database) was run previously.

When the Database icon with the exist command is executed, IconAuthor checks your database file directory. If the database file is found, the logical constant .T. is stored in the system variable @_FOUND. If the database file is not found, .F. is stored in @_FOUND.

Structure your application to take different branches depending on the value in @_FOUND. If the database does not exist, execution should flow to a Database icon that creates it or displays a message explaining that the database was not found. If the database exists, execution should flow to a Database icon that makes it the current, usable database.

Warning: It is important that the application does not create an IconAuthor database that already exists. If you create a database with a filename that already exists, all the data it contained is erased.

Related Topic:

Database Commands

create command

Syntax: **create *database_filename* with *format_filename***

Use the create command to create a new database file. You specify both the name you want to assign to the new database file and the name of the format file on which its structure should be based. When the Database icon is executed, IconAuthor searches your format file directory for the specified format file and creates the new database file in your database file directory. Format files have .FMT extensions and database files have .DBF extensions.

Warning: It is important that the application does not create a database that already exists. If you create a database with a filename that already exists, all the data it contained is erased.

When a database file is created it is not automatically made current and available for use. To make it available, add another Database icon to issue a use command.

Related Topic:

Database Commands

use command

Syntax: **use *database_filename***

The use command makes an existing database file the current database file. When a database file is current, subsequent Database commands that search for data or add records to the database are directed to this database file.

Only one database file can be current at any given time. It is used until another use command is issued to make another database file the current database.

Example: use ROSTER.DBF

The use command makes ROSTER.DBF the current database file

Related Topics:

Database Commands

add command

Syntax: **add**

The add command creates a new blank record at the bottom of the current database. The new record becomes the current record. The add command invalidates the last locate command because it moves to the end of the file.

Related Topics:

Database Commands

replace command

Syntax: **replace *field_name* with *value***

The replace command acts on the current record, by replacing the data in the specified field. Prior to this command, an add, locate, or next command is used to make a record current.

The value you specify must be of the same type as the field in which it is being stored. For example, if the field is GRADE, which is a numeric type field, only a numeric value can be stored in it. If the field is PASS, which is a logical type field, only a .T. or an .F. can be stored in it.

Note: Do not use the replace command after a use or delete command. The current record is unknown, and the replace command may cause the wrong data to be modified.

Related Topics:

Database Commands

locate command

Syntax: **locate *condition***

The locate command searches the database from the beginning, trying to find the first field whose value meets a specific condition. If the value is found, the record in which the value is found is made the current record.

Typically, once a record is made current by a locate command, it is manipulated using another database command. For example, a replace command places new data in one of the fields in the located record.

After a locate command, an application might also use a next command to search down through the database looking for the next field whose value meets the condition expressed in the original locate command. The next command can be used repeatedly.

Each time a locate command (or a next command) is executed, the system variable @_FOUND is set to either .T. or .F. If a value is found that meets the condition, @_FOUND is set to .T. and that record is made current. If a value is not found, @_FOUND is set to .F. and there is no current record.

Structure your application so that it takes one branch versus another depending on the value in @_FOUND. If a record is found, that contains an appropriate value, the application should act on the value or another value in that record. If a record is not found, a message can be displayed that indicates that a record was not found.

The locate command is always used with an expression that describes the condition you are trying to find. The expression must include the name of the field being searched and can contain one or more operators, constants, and/or variables.

Note: When you type an expression, be sure to use a space before and after any operator.

The following relational operators can be used in expressions:

=	equal to
<=	less than or equal to
<	less than
>	greater than
>=	greater than or equal to
<> or #	not equal to

The following logical operators can be used in expressions:

.AND.	both conditions are true
.OR.	at least one of the conditions is true
.NOT.	the negative of the condition is true

Related Topics:

Database Commands

next command

Syntax: **next**

The next command is a continuation of the locate command. The locate command searches the database from the beginning, trying to find the *first* field whose value meets a specific condition. If the value is found, the record in which the value is found is made the current record. A subsequent Database icon with a next command continues to search down through the database, for the *next* record that satisfies the condition in the original locate command.

Often, once a record is made current by a locate or a next command, it is manipulated using another database command. For example, a record is made current and then a Database icon with a replace command replaces a value in one of the fields of the current record.

Each time a next command (or a locate command) is executed, the system variable @_FOUND is set to either .T. or .F. If a value is found that meets the condition, @_FOUND is set to .T. and that record is made current. If a value is not found, @_FOUND is set to .F. and there is no current record.

Structure your application so that it takes one branch versus another depending on the value in @_FOUND. If a record is found, that contains an appropriate value, the application should act on the value or another value in that record. If a record is not found, another branch can be taken, such as one where a message is displayed that indicates that there are no more matching records to be found.

Related Topics:

Database Commands

store command

Syntax: **store *field_name* to *variable***

After a record is made current (with either a locate or next command) the store command is used to store a piece of data from the specified field in the specified IconAuthor variable. If the variable does not already exist, the store command creates it.

Note: Do not use the store command after a use or delete command. The current record is unknown, and the store command may cause the wrong data to be modified.

Related Topics:

Database Commands

delete command

Syntax: **delete**

After a record is made current with either a locate or next command, the delete command deletes the current record from the database.

After a delete command, the last locate command is no longer valid. Therefore, you cannot use a next command to find the next record that meets the previously specified condition. The locate command must be given again.

After a delete command, the value of the system variable @_FOUND is .T. if there is another record in the database. If the last record has been deleted and the database is empty, the value in @_FOUND is .F.

Related Topics:

Database Commands

close command

Syntax: **close**

The close command closes the current database.

Note: For dBASE III Plus and dBASE IV users, when a database is closed, any records that were marked DELETE, are physically removed from the file. Any records that were marked for deletion through the dBASE application are also deleted. The IconAuthor close command automatically does a dBASE PACK command.

Related Topics:

Database Commands

Date&Time Icon

The Date&Time icon stores current time information for use in your application. You can store the current date, the current time, or the time elapsed since the start of the application.

Suggested Uses:

- Inform the user of the current date and/or time at some time during the course of execution, for example at login.
- Store the current date in a database.
- Measure how long it takes a user to complete a course.
- When developing an application, measure how long it takes an event to complete (compare the effectiveness of using one type of structure for a task versus another).

Content Editor Text Boxes:

Time Variable
Type

Time Variable

Specify the name of the variable in which you want to store the date/time information.

Drop-down List Box Items:

Variable Selector

Type

Specify the type of date or time information you want to store in the Time Variable text box. Acceptable values are: **date**, **dbdate**, **elapsed**, **time**, or a variable.

Drop-down List Box Items:

date - Stores the current date. Uses the format Fri Mar 10 1991

dbdate - Stores the current date. Uses the format yyyymmdd, used by the IconAuthor Database icon. March 10, 1991 is 19910310

elapsed - Stores the whole number that represents the elapsed time in seconds since the start of the application

time - Stores the local time in the format hh:mm:ss. An example of this format is 18:22:35.

Variable Selector

Display Icon

Use the Display icon to create one of several kinds of screen displays. Each Display icon you use can display a bitmap graphic, an animation script, or a SmartObject file. The Display icon is also used to pre-load graphics into memory which will increase the speed at which they are displayed.

Suggested Uses:

- Display a SmartObject file that contains live objects with which the user can interact. Some examples of live objects are Push Buttons, Audio Buttons, Movie objects, Text objects, and List Boxes. (Audio objects play MIDI, Wave audio, or CD-audio and Movie objects play digital video or third party animations.)
- Use two Display icons to display a graphic file and then a SmartObject file with text, on top of it. The SmartObject file has a transparent background so it appears superimposed on the graphic. This technique, of creating text and graphics in separate files, lets you use the graphic file repeatedly with different text. For example, your application can use a graphic of a menu made of buttons several times with different SmartObject files.

  Preload multiple graphic files into memory and optimize the speed at which they are displayed.
- Enhance courses and presentations with animation scripts.

Content Editor Text Boxes:

File Type
File Name
Location
Parameters

File Type

Specify the type of file you want to display. Acceptable values are: **animate**, **bitmap**, **SmartObject**, or a variable.

Drop-down List Box Items:

animate - Indicates that you want to run an IconAnimate animation script.

bitmap - Indicates that you want to display a bitmap graphic, such as a file created with Paintbrush. Select this item if you are displaying a graphic with one of the following formats:

.ATT	.FIF	.KFX	.PSD	.XPM
.BMP	.GIF	.LV	.RAS	.XWD
.CAL	.GX2	.MAC	.RLE	
.CLP	.ICA	.MSP	.TGA	
.CUT	.ICO	.PCD	.TIF	
.DCX	.IFF	.PCT	.WMF	
.DIB	.IMG	.PCX	.WPG	
.EPS	.JPG	.PIC	.XBM	

SmartObject - Indicates that you want to display a SmartObject file, created with the SmartObject Editor. The file extension is .SMT

Variable Selector

Filename

Specify the name of the file you want to display. Acceptable values are a variable or a filename that has one of the following formats:

.ATT	.FIF	.KFX	.PSD	.XPM
.BMP	.GIF	.LV	.RAS	.XWD
.CAL	.GX2	.MAC	.RLE	
.CLP	.ICA	.MSP	.TGA	
.CUT	.ICO	.PCD	.TIF	
.DCX	.IFF	.PCT	.WMF	
.DIB	.IMG	.PCX	.WPG	
.EPS	.JPG	.PIC	.XBM	

Drop-down List Box Items:

Directory
SmartObject Editor
Animation Editor
Variable Selector

Location

Specify the location of the file you want to display. The Location text box actually lets you choose coordinates for the *upper left corner* of the file being displayed. If the file is smaller than the full screen, you can choose coordinates that place the file anywhere on the screen. The default, 0,0 places the file in the upper left corner of the screen. You can use other coordinates, such as "centered, centered" that center the file; or have it appear in a particular corner.

If you choose negative coordinates for upper left corner of the file, or extremely large coordinates, part or all of the file may not appear on the screen. Negative coordinates may place the file very high (off the screen) or to the left (off the screen). Large coordinates may place the file very low (off the screen) or to the right (off the screen).

Acceptable values are a pair of screen coordinates or a variable.

Drop-down List Box Items:

Location Editor
Variable Selector

Parameters

Use this text box to assign parameters to the file being loaded or displayed.

Acceptable values are a SmartObject page, a display effect, a load command, or a variable.

Drop-down List Box Items:

load - Pre-loads graphic.
load discardable - Pre-loads graphic and removes from memory immediately after display.
remove - Removes graphic from memory.
remove all - Removes all graphics from memory.
Effect Selector - Lets you select a special effect to use when displaying a graphic file.
Page List - Lets you select the SmartObject page you want to display.
Variable Selector

This text box performs differently depending on the kind of file you are displaying. Acceptable values: are a SmartObject page (for a SmartObject file), a display effect (for a graphic file), a pre-load command, or a variable.

Displaying a SmartObject Page:

If you are displaying a page in a SmartObject file, use this text box to specify the page name. To select the appropriate page name, choose Page List from the drop-down list box. A dialog box appears which lists all of the saved pages created under the filename specified in the Filename text box. Double-click on the page you wish to display.

Or, you can use a variable in this text box. Type a variable name or choose Variable Selector from the drop-down list box.

Displaying a Graphic File:

If you are displaying a bitmap graphic file, specify the special effect you want to use. IconAuthor allows you to specify a single effect or a double effect. A single effect occurs in one screen pass. A double effect occurs in two screen passes. When a double effect is used, on the first pass, the new graphic is merged with the existing graphic. On the second pass, the new graphic replaces the existing graphic. To specify the effect, click on the drop-down arrow of the Parameters text box and choose the Effect Selector.

Preloading a Graphic File:

In addition to displaying graphic files, you can use the Display icon to load a graphic into memory. If you are loading a graphic which you will use repeatedly, use the load command. The load command keeps the graphic in memory until either a remove command is issued, or the application finishes. The number of graphics you can load into memory is limited only by the amount of memory in your (and your end-user's) computer. If you are loading a graphic which will not be used repeatedly throughout the application, use the load discardable command. When the load discardable parameter is used, you are telling IconAuthor to remove the graphic, after it is displayed, to make room for the loading of another graphic. The remove command will remove (unload) the graphic file specified in the Filename text box from memory. Remove all will remove all graphic files from memory.

The drop-down list box contains frequently used special effects and the Variable Selector.

Ellipse Icon

The Ellipse icon dynamically draws an ellipse.

You can control whether the ellipse is filled and outlined, or outlined only. By default, the outline color is black and the fill color is white. To create an ellipse using alternative colors, precede the Ellipse icon with a Color icon. The fill and outline colors you specify are in effect until you use another Color icon. (If you use the Color icon to specify a fill color of TRANSPARENT, the underlying screen display will show through the interior of the ellipse.)

Suggested Uses:

- Create an ellipse that is a border around part of the screen
- Mask an ellipse-shaped area of the screen

Content Editor Text Boxes:

Upper Left Corner
Lower Right Offset
Line Width
Filled or Outline

Upper Left Corner

To position the ellipse, you actually describe the imaginary rectangle that surrounds the ellipse. Use the Upper Left Corner text box to position the ellipse on the screen by specifying the location of the upper left corner of the imaginary rectangle. Acceptable values are: **upper left**, a pair of screen coordinates, or a variable.

Drop-down List Box Items:

upper left - Defines the upper left corner of the imaginary rectangle as the upper left corner of the screen.

Area Editor

Variable Selector

Lower Right Offset

Specify the width and height (in pixels) of the imaginary rectangle that surrounds the ellipse.

Note: A value automatically appears in this text box if you use the Area Editor to specify the value in the Upper Left Corner text box.

Acceptable values are: **lower right**, two numbers that represent the width and height respectively (separated by a comma), or a variable.

Drop-down List Box Items:

lower right - Defines the width and height of the imaginary rectangle so that its lower right corner is located at the lower right corner of the screen.

Variable Selector

Line Width

Specify the width (in pixels) of the border of the ellipse. If you are drawing an outlined ellipse, use a line width of 1 pixel or more. If you are drawing a filled ellipse, use a line width of 0 pixels or more. (If the line width of a filled ellipse is equal to 0, it has no border and uses only a fill color.)

Acceptable values are: any whole number (including 0) or a variable.

Drop-down List Box Items:

assorted whole numbers - Frequently used line widths.

Variable Selector

Filled or Outline

Indicate whether the ellipse is filled and outlined, or outlined only.

Acceptable values are: **filled**, **outline**, or a variable.

Drop-down List Box Items:

filled - Creates a filled ellipse. If the line width of a filled ellipse is greater than 0, it uses a fill color and an outline color. If the line width of a filled ellipse is equal to 0, it has no border and uses only a fill color.

outline - Creates an outlined ellipse. Do not use a line width of 0 to create an outlined ellipse.

Variable Selector

Exit Icon

The Exit icon is key to controlling the flow of execution in your application. Depending on how you use the Exit icon, it can cause execution flow to exit from one of the following types of structures:

- a composite icon, such as a Menu icon, or a composite that you create
- a loop
- a series of 2 or more nested loops
- a Help application
- a subroutine
- a sub-application
- a main application
- Windows

Content Editor Text Boxes:

Exit From
Return List

Exit From

Specify the type of structure being exited.

Acceptable values are: **application**, **windows**, **composite**, **help**, **subapp**, **loop**, **subroutine**, any number greater than 2, or a variable.

Drop-down List Box Items:

application - Causes an exit from the entire application. Even if this kind of exit occurs from within a Help application (that was accessed from the main application), both the Help application and the main application are exited.

composite - Causes an exit from a composite. If the Exit icon is executed, execution flows to the icon below the lead icon in the composite.

Windows - Causes an exit from the application and Windows.

subapp - Causes an exit from a sub-application that was called from the main application. If the Exit icon is executed, execution returns to the main application from which the sub-application was called. The main application resumes execution with the icon below the SubApp icon.

help - Causes an exit from a Help application. If the Exit icon is executed, execution returns to the main application from which Help was accessed. Execution resumes with the icon that was current when Help was accessed.

loop - Causes an exit from a loop (a composite Loop icon or a composite LoopIndex icon). If the Exit icon is executed, execution flows to the icon below the lead icon in the loop composite.

subroutine - Causes an exit from a subroutine. The main application resumes execution with the icon below the Subroutine icon.

2 - Causes an exit from two loops. This means that the exit occurs from within the inner (second level) loop and execution flows to the icon below the lead icon in the outer (first level) loop composite.

Note: The Exit From text box can contain any positive integer. The result is an exit from the specified number of nested loops.

Variable Selector

Return List

Use the Return List text box if you specified subroutine in the Exit From text box.

When a Subroutine icon causes a subroutine to execute, parameters can be passed to the subroutine. In turn, when the subroutine is exited, it can pass values back to the main application. The Exit icon in the subroutine passes one or more fixed values and/or values stored in variables, back to the Subroutine icon (that initially called the subroutine) in the main application. When the values are returned to the Subroutine icon they are assigned to variables.

Use the Return List text box of the Exit icon to specify the names of the values and variables you want to pass back to the main application. The values of the variables you list in the Return List text box of the Exit icon are assigned, in the order they are listed, to the variables listed in the Subroutine icon.

The variable values returned to the main application can be single value variables or indexed variables. If the variable being returned is indexed, then the receiving variable becomes an indexed variable. For example, the value of the variable @LETTERS is returned to the main application. @LETTERS is an indexed variable with three elements (@LETTERS[1]=A, @LETTERS[2]=B, and @LETTERS[3]=C). The variable listed in the Receive List text box of the Subroutine icon also becomes an array. If the Receive List text box variable is @VAR, then @VAR[1]=A, @VAR[2]=B, and @VAR[3]=C.

Acceptable Values are: any positive or negative decimal or integer number, or a variable.

Drop-down List Box Item:

Variable Selector

Related Topics:

Subroutine Icon

Font Icon

The Font icon lets you access the Font Editor dialog box to set the font, style, size and color of the characters generated by the Text icon (an ASCII file) and the Write icon (dynamically generated text). It also lets you control the font, style, size and color of characters displayed when a user types a response to an Input icon. The Font icon does not control the display of text within SmartObject or Graphic files.

If you do not use a Font icon, the smallest available size of the System type font is in effect for text displays. The fonts that are currently loaded in your windowing system are the fonts (and sizes) that are available for use with IconAuthor.

Choose Font Editor... from the drop-down list to access the Font Editor. This is the same Font Editor that appears at other times in IconAuthor except that there are two extra fields: Escapement and Orientation.

Escapement

Specify the angle at which you want each entire text line to appear. This feature is only available for vector type fonts. Examples of vector fonts are Modern, Roman and Script.

Orientation

Specify the angle at which you want each character in a text line to appear. This feature is also only available for vector type fonts.

Help Icon

Use the Help icon to allow users to use a Help application (online Help) while they use your application. The Help icon specifies the name of the Help application the user can access, and the action the user must take in order to access the Help application.

Important: Do not use this icon if your application uses live objects (created via the SmartObject Editor.)

Place the Help icon as the first icon in the main application to initialize the help function. When you enter information in the Content Editor text boxes, you indicate 1) the name of the Help application, and 2) how the user accesses Help.

Content Editor Text Boxes:

Help Filename
Help Escape Type
Help Escape
Help Variable (reserved for future use)

Related Topics:

Creating a Help Application

Help Filename

Specify the name of the Help application. The Help application should have a .IW extension like any other IconAuthor sub-application.

Only one Help application can be active at one time.

Acceptable values are: a filename with a .IW extension or a variable.

Drop-down List Box Items:

Directory
Variable Selector

Help Escape Type

Specify the kind of action the user must perform to execute (escape to) the Help application.

There are three general kinds of actions you can let users perform to access Help. They can press a key, click on an area of the screen, or touch an area of the screen (if they are using a touch screen).

Regardless of the kind of action you let users perform, they can only press a key, click on the screen, or touch the screen when IconAuthor is waiting for input. When an Input icon or an InputMenu icon is executed, IconAuthor is waiting for a response from the user. It is at this point that the user can access Help. A user cannot access Help when IconAuthor is executing icons. For example, Help is not accessible while an animation script is running or while a graphic is being slowly displayed on the screen.

Acceptable values are: **all mouse**, **char**, **L mouse down**, **R mouse down**, **touch**, or a variable

Drop-down List Box Items:

all mouse - The user clicks any mouse button to execute the Help application.

char - The user presses any single character key, such as F1 to execute the Help application.

L mouse down - The user presses the left mouse button to execute the Help application.

R mouse down - The user presses the right mouse button to execute the Help application.

touch - The user touches an area of an active touch screen to execute the Help application.

Variable Selector

Help Escape

Specify the key or screen area that the user presses, clicks on, or touches, to execute (escape to) the Help application.

The Help Escape must match the Help Escape Type. For example, if the Help Escape Type is L mouse down, the Help Escape must be a selectable area of the screen. If the Help Escape Type is char, the Help Escape must be a key such as F1.

If you specify a key or an area of the screen as the Help Escape, it is reserved for the entire application and cannot be used for any other purpose. For example, if you specify F1 as the Help Escape key, it cannot be used for any other purpose within the main application.

If you have chosen a key as the Help Escape, it is helpful to have a reminder on each display that appears when IconAuthor is waiting for input, such as "Press F1 for Help".

If you have chosen an area as the Help Escape, it is helpful to have a button labeled Help on each display that appears when IconAuthor is waiting for input.

Note: If you select an area of the screen as the Help Escape, the user clicks on it to access the Help application. It is *not* necessary to redefine that area as "hot" or "selectable" at another point in the application. When you use an InputMenu icon to make several areas of the screen input selectable, do *not* include the Help Escape area. It is already selectable.

Acceptable values are: 4 numbers, separated by commas, that define a rectangular area on the screen, the name of a key (such as F1), or a variable. If you specify 4 numbers, the first two digits are the x,y coordinates of the upper left corner of the area, the second two digits are the width and height of the area.

Drop-down List Box Items:

Area Editor
Variable Selector

Creating a Help Application

The Help icon should be the first icon in your main application. When the application runs, the help function is initialized.

Note: Only one Help application can be active at a time. If another Help icon is executed later in the main application, it replaces the previous Help icon.

The first icon in a Help application is a Snapshot icon. The Snapshot icon saves the context of the main application when the Help application is first called. The context is restored when control is passed back to the main application. (For more information see "Snapshot icon".)

You can make your Help application "context sensitive" by using variables. Context sensitive help provides the user access to different information depending on what part of the main application they are using when they access Help.

For example, an application can contain two modules, a tutorial and a test. At the beginning of the tutorial, a Variable icon sets @HELP_BRANCH equal to 1. At the beginning of the test, another Variable icon sets @HELP_BRANCH equal to 2. When the user accesses help, the Help application uses a composite Branches icon to test whether @HELP_BRANCH is equal to 1 or 2. If @HELP_BRANCH is equal to 1, the first branch is executed and the user accesses information on how to use the tutorial. If @HELP_BRANCH is equal to 2, the second branch is executed and the user accesses information on how to use the test.

To give the user an opportunity to return to the main application, use one or more Exit icons in your Help application. Specify "help" in the Exit From text box of the Exit icon. When an Exit icon is executed, execution flow returns to the main application at the point at which Help was accessed. The Input icon or an InputMenu icon that was awaiting a response (when the user accessed the Help application) is still the current icon. The Snapshot icon at the beginning of the Help application restores the screen context when control is returned to the main application.

Note: If you specify "application" in the Exit From text box of the Exit icon, both the Help application and the main application are terminated.

If Icon

The If icon controls the execution flow of an application. Because the If icon forms a type of branch in the structure, unlike most other icons, you can build or paste an icon to the right of the If icon.

The If icon compares two values and checks whether a condition is true or false. For example, it can check whether two values are equal. If they are, the icon below the If icon (the true path) is executed. If they are not, the icon to the right of the If icon (the false path) is executed *and* then the icon below the If icon (the true path) is executed.

When an Exit icon is built into the false path of an If icon, the false path of the If icon can be executed, without being followed by the true path.

Note: A single If icon behaves differently from the If icons that are contained in the composite Branches icon. In an If icon within a Branches composite, if the condition is true, the true path is executed. If a condition is false, *only* the false path is executed.

You can take a single If icon, and build it into the existing series of If icons in Branches composite. However, when you add an If icon in this way, it performs in the same manner as the four If icons that are initially part of every Branches composite.

The single If icon is well suited to some very specific tasks. As an example, it is particularly useful to check the existence of a database before attempting to recreate it.

Content Editor Text Boxes:

Condition 1
Test
Condition 2
Condition Type

Related Topics:

Comparing Numbers
Comparing Non-Displayable Characters
Character String Matching

Condition 1

Specify the name of a variable that you want to compare to the value in the Condition 2 text box.

Drop-down List Box Items:

Variable Selector

Test

Specify the test that you want to use to compare the value in the Condition 1 text box and the value in the Condition 2 text box.

Acceptable values are: **EQ**, **NE**, **LT**, or **GT**

Drop-down List Box Items:

EQ - Tests if Condition 1 is *equal to* Condition 2.

NE - Tests if Condition 1 is *not equal to* Condition 2.

LT - Tests if Condition 1 is *less than* Condition 2.

GT - Tests if Condition 1 is *greater than* Condition 2.

Condition 2

Specify the value to which you want to compare the variable in Condition 1.

Acceptable values are: a numeric value (including negative and decimal numbers), an alphanumeric character, an alphanumeric string, a non-displayable key (such as the RETURN key), or the coordinates of a rectangular area.

Drop-down List Box Items:

- Area Editor
- Object Name Selector
- Object Event Selector
- Variable Selector

Condition Type

Indicate kind of data being compared.

Acceptable values are: **alphabet**, **numeral**, **rectangle**, or a variable.

Drop-down List Box Items:

alphabet - Indicates that the values you are comparing are alphanumeric characters.

numeral - Indicates that the values you are comparing are numeric.

rectangle - Indicates that the values you are comparing are the rectangle coordinates.

Variable Selector

Input Icon

The Input icon allows the user to use a keyboard, mouse, or touchscreen to provide information to the application. For example, an Input icon can let a student type an answer to a question, or allow a customer to enter a name, an address, a telephone number, or a date.

Important: Do not use the Input icon if you are using live objects in your application. Live objects are displayed via SmartObject files. SmartObject files also allow user input, but use different methods and different icons.

When you add content to an Input icon, you provide information such as the name of the variable in which the user's response is stored, the kind of input the user is allowed to provide, and whether there is a time limit on the user's response.

The icon below the Input icon is not executed until either the user makes a response or a specified time-out period expires. When an Input icon is executed, key information is stored in the system variables @_USERTIME, @_TIMEOUT and @_SELECTION.

Content Editor Text Boxes:

Variable Name
Input Type
Time Limit

@_USERTIME

@_USERTIME keeps track of the amount of time (in hundredths of a second) a user takes to respond to an Input or InputMenu icon. This can be useful information if your application is a test and you want to know how long it takes a user to answer each question.

@_TIMEOUT

@_TIMEOUT keeps track of whether a user responds to an Input or InputMenu icon before timeout occurs. When the user responds before timeout, @_TIMEOUT = 0. When the user does not respond before timeout, @_TIMEOUT = 1.

The information stored in @_TIMEOUT is particularly important if a user response does not occur because a system is left unattended. If a user does not respond, you may want execution to automatically flow back out to a main menu.

Regardless of whether the user responds to the input screen, when the timeout occurs, execution flows to the next icon. Below the Input icon, use an If icon to test the value in @_TIMEOUT. If @_TIMEOUT = 0, the user responded before timeout and execution can flow to the next part of the structure. For example, after this icon, another input icon may request further information, or a Branches composite may evaluate the user's response. If @_TIMEOUT = 1, the user did not respond before timeout. A branch can be executed that causes an exit to occur from this part of the application.

@_SELECTION

@_SELECTION stores the terminate key used to end input.

Variable Name

Specify the name of the variable in which the user's input is stored.

Drop-down List Box Item:

Variable Selector

Input Type

Specify a string of parameters that defines the type of input the user can provide.

Hint: Rather than constructing this string by typing in the blank Input Type text box, use the Input Selection Editor to generate the string automatically.

Acceptable values are: a default string of parameters, a custom string of parameters, or a variable.

Drop-down List Box Items:

Input Selection Editor
Variable Selector

Related Topics:

Using the Input Selection Editor
Input Icon

Time Limit

Specify whether or not there is a limit to the amount of time a user can take to respond to the input screen. If you do not want a time limit, specify 0 or leave the text block blank. If you do want a time limit, enter a numeric value.

If you enter a time limit and the user does not provide input within the specified time, the icon below the Input icon is executed. When the Input icon is executed, two system variables are set. @_TIMEOUT = 1 if timeout occurred before the user responded. @_TIMEOUT = 0 if the user responded before timeout. Also, the @_USERTIME contains the number of one-hundredths of a second it took the user to respond.

Acceptable values are: any positive real number (such as 10 or 10.5) or a variable.

Drop-down List Box Item:

Variable Selector

Using the Input Selection Editor

The Input Selection Editor appears when you choose that item from the drop-down list box of the Input Type text box. Select the general category of input you want the user to enter: Text, Mouse, Numeric, or Date.

Text - The user enters a string or a character using the keyboard.

Mouse - The user selects a single point on the screen using a mouse or touchscreen.

Numeric - The user enters a numeric value using the keyboard.

Date - The user enters a date.

To accept the default parameters for the selected input type, choose OK. The dialog box is closed, and the default parameters are automatically entered in the Input Type text box.

To create a custom parameter string for the selected input type, choose Options...

A different dialog box appears depending on the input type you are defining: the Text Input dialog box appears for text input, the Mouse Input dialog box appears for mouse input, and so on.

Related Topics

Using the Text Input Dialog Box
Using the Mouse Input Dialog Box
Using the Numeric Input Dialog Box
Using the Date Input Dialog Box

Using the Text Input Dialog Box

When you select Text and choose Options... from the Input Selection Editor dialog box, the Text Input dialog box appears and contains the following components:

Result box
Initialize Input Variable check box
Display area
Format area
Input Termination area
Options >> button

Result Box

Initially, the default parameter string for the general input type you selected appears in this box. As you change parameters by making different selections, the new parameters appear in this box.

When you close the dialog box, the parameters in the result area are returned to the Input Type text box from which the Input Selection Editor was accessed. Each parameter is separated from the next by a semicolon.

The result box is scrollable from left to right. To view hidden parameters, click inside the box and use either the right or left arrow key, the Home key or the End key to see another part of the string.

Related Topics:

Using the Text Input Dialog Box
Using the Mouse Input Dialog Box
Using the Numeric Input Dialog Box
Using the Date Input Dialog Box

Initialize Input Variable check box

Specify whether the Input Variable is initialized when the input screen appears. (X = initialize, blank = do not initialize). If you do not initialize the input variable and it contains a value, that value appears on the screen, and the user has the option of editing it.

If you initialize the input variable, the input area on the screen is blank. The user must enter new input that will be stored in the input variable.

Related Topics:

Using the Text Input Dialog Box
Using the Numeric Input Dialog Box
Using the Date Input Dialog Box

Display area

Specify whether the input is displayed on the screen.

On - Turns input display on. All displayable characters entered from the keyboard are echoed on the display screen during input.

Off - Disables input display. Key input is stored in the memory buffer only.

Masked - Masks the input being displayed. The default mask character is #. The mask character is displayed in place of the key pressed from the keyboard during input.

Location text box

Specify the starting location of input on the screen. Display Location is only available if Display Input is set to On or Masked.

upper left - Defines the starting location of input as the upper left corner of the screen.

Location Editor

Variable Selector

Caret text box

Specify the shape of the cursor that is displayed during input. The choices are:

underline - A flashing underline appears where characters or numbers will appear. As the user types, the underline is pushed to the right.

vertical line - A flashing vertical line appears where characters or numbers will appear. As the user types, the line is pushed to the right.

solid block - A flashing block appears where characters or numbers will appear. As the user types, the block is pushed to the right.

grey block - A flashing grey block appears where characters or numbers will appear. As the user types, the block is pushed to the right.

none - There is no cursor as text is entered.

Note: If you choose not to display the input on the screen, the cursor you select in the Caret text box is disregarded.

Related Topics:

Using the Text Input Dialog Box
Using the Numeric Input Dialog Box
Using the Date Input Dialog Box

Format Area

Style text box

Specify the kind of characters that the user can enter from the keyboard.

Standard - This is the default. Accepts all characters, numbers, and symbols from the keyboard in either uppercase or lowercase.

Uppercase - Accepts all characters, numbers, and symbols. Converts all characters (a through z) to uppercase. The conversion occurs before the character is displayed on the screen. This means that although the user may press "a," an "A" appears on the screen.

Lowercase - Accepts all characters, numbers, and symbols. Converts all characters (A through Z) to lowercase. The conversion occurs before the character is displayed on the screen. This means that although the user may press "A," an "a" appears on the screen.

Logical - Accepts only a logical .T. or .F. entry. The user enters "T" or "F" and the results are stored with a period before and after the character. If the character is entered in lowercase, the system converts it to uppercase before the results are displayed on the screen. If the user tries to enter an invalid character, the system beeps.

Y or N - Accepts only "Y" or "N" from the keyboard. The user can type the input in uppercase or lowercase. If the character is entered in lowercase, the system converts it to uppercase before the results are displayed on the screen. If the user tries to enter an invalid character, the system beeps.

String Length text box

Specify the maximum number of characters that the user can enter. If the user attempts to enter more than the allowable number of characters, the system beeps.

The default value is 1, which indicates single character input. The range is 1 to 256. This field is disabled if the Logical or Y/N text formats are selected because Logical and Y/N formats require only 1 character.

Picture text box

Predefine the specific kinds of characters that are allowed for each position in an input field. For example, if the input is intended to let the user enter a zip code, you can specify that the input field only accepts five numbers. The number of characters you specify as part of the Picture must match the number you specify for the Maximum Length of Text. If the user tries to press a terminate key before entering the number of characters in the picture, the system beeps.

Use the following characters to create a picture:

9 - Numeric characters only (0 - 9)
Z - Numeric characters, leading zeros, and blank positions
A - Alphabetical characters only (a - z, A - Z)
X - Alphabetical or numeric characters (a - z, A - Z, 0 - 9)
S - Any alphabetical or symbol (including spaces)
B - Any alphabetical or numeric character, or symbol (including spaces)

Any other characters you specify as part of the picture become part of the input.

Related Topic:

Using the Text Input Dialog Box

Input Termination Area

Confirm Input check box

Specify whether the user confirms that input is complete before execution flows to the next icon. If this option is toggled on, the user has to press one of the keys specified in the Terminate Key box to proceed. If this option is toggled off, the user enters the character or string, and execution automatically flows to the next icon when the maximum number of characters have been entered.

Allow Null check box

Specify whether blank input entries are allowed. This feature is useful when you want to allow the user to bypass entering input.

If this option is on, the user can press a terminate key and execution flows to the next icon. If this option is off, the user must enter a character or string before execution can flow to the next icon.

Termination Keys box

Specify the keys you want the user to be able to press to indicate that they are finished providing input. You can specify as many as 25 keys or multi-key combinations. The default is the RETURN key.

There are two ways to enter acceptable terminate keys into the Terminate Key box:

- Click on the Record button to automatically record keys and multi-key combinations as you press them. Once the Record button is selected, its label changes to Stop. Each key or key combination is entered into the list box as it is pressed. When you are finished defining terminate keys, click on the Stop button (its label changes to Record). You cannot close the Text Input dialog box while the button is labeled Stop.
- Type the terminate key or multi-key combinations into the list box.

The following keys are available as single key selection labels:

A through Z keys	HOME (Home key)
0 through 9 keys	INS (Insert key)
0 through 9 (keypad)	TAB (Tab key)
BS (Back Space key)	NEXT (Page Down key)
DEL (Delete key)	PRIOR (Page Up key)
END (End key)	LEFT (left arrow key)
RETURN (Enter key)	RIGHT (right arrow key)
ESC (Esc Key)*	UP (up arrow key)
F1... F12 (F1 through F12)	DOWN (down arrow key)
	PAUSE (pause/break key)

* The Esc key is used in the IconAuthor Authoring system to escape from a running application and return to the IconAuthor window. However, you can use the Esc key in your applications because Esc is interpreted as a standard non-displayable character by the IconAuthor Presentation system. The only case in which Esc is not available for use is if it is the designated break key for the Presentation system. To determine which key or multi-key combination is reserved as the break key for the Presentation system, look at the BreakKey= entry in the IconAuthor section of your PRESENT.INI file.

To specify a multi-key combination for a selectable area use one or any combination of SHIFT, CONTROL, or ALT followed by one of the valid single keys described previously. Use a hyphen to separate one key from another if they are all part of a multi-key combination. For example, CONTROL-SHIFT-F7, or ALT-F4 are valid selection labels.

When the user presses a terminate key, the value of the terminate key is stored in the system variable @_SELECTION. For example, if the user enters a string and presses the Delete key, @_SELECTION = DEL. The value in @_SELECTION can be evaluated in a subsequent icon to determine the next sequence of events in the application. For example, you might want to evaluate the value in

@_SELECTION so that TAB lets the user proceed to the next input field, and PRIOR lets the user work with the last input field.

Related Topics:

Using the Text Input Dialog Box
Using the Numeric Input Dialog Box
Using the Date Input Dialog Box

Options>> Button

When you choose Options>>, the input dialog box is extended and you can create expressions in the validation area to validate the user's input before it is accepted. If the user enters a response that is not valid, the system beeps, and the user must enter a new value.

Validation view box - The validation view box is the large box at the top of the validation area used to create the validation expressions. There are two ways to create an expression in the validation view box. You can type an expression directly into the validation view box. Or, you can use the features in the bottom part of the validation area to automatically generate the parts of the expression in the validation view box. If the expression you are entering is longer than one line, the expression automatically wraps to the next line. Do not attempt to include carriage returns.

Input Variable button - Within an expression, the value in the input variable is compared with one or more strings or numeric values represented as constants or variables. The input variable is always represented by @@. Click on the Input Variable button to automatically insert an @@ in the validation view box.

operator area - The operator area contains 16 buttons that represent the operators necessary to create an expression. Click on an operator button to automatically insert that operator (=, +, etc.) in the validation view box.

freetype entry box - The freetype entry box is the text box below the validation view box. When you want to include a string, number or a variable as part of an expression, type it into the freetype entry box and press Return. The number or variable is automatically inserted in the validation view box with surrounding quotes if necessary, and a space before and after.

Note: When you use the buttons and freetype entry box to create an expression, IconAuthor automatically inserts the necessary space before and after the various parts of the expression. When you type an expression directly into the validation view box, remember to include spaces as necessary, for example, before and after an operator.

Related Topics:

Text Validation Rules
Numeric Validation Rules
Date Validation Rules
Using the Text Input Dialog Box
Using the Numeric Input Dialog Box
Using the Date Input Dialog Box

Text Validation Rules

When you choose Options>> from the Text Input dialog box you can create an expression in the validation area. The expression you create evaluates the user's input to determine whether it is acceptable. As an example, consider the case where the user is answering the question, "What screen color do you want?" You include the following expression in the validation view box:

@@ $ "RED,BLUE,GREEN"

When the user types a response, the input is validated according to this expression. If the input is equal to RED, BLUE, or GREEN, it is accepted. If it is some other value, the system beeps and the user must re-enter input.

After you enter all of the input validation information, choose OK. The validation area is removed. Choose OK again to close the remaining portion of the Text Input dialog box.

Character Operators

In the validation view box, you can specify an expression that uses one of the following character operators to manipulate character strings:

& operator
EXTRACT(start,len,str) operator
UPPER(str) and LOWER(str) operators

Relational Operators and Character Strings

You can specify a validation expression that uses a relational operator to compare character strings.

Operator	Meaning
<	less than
<=	less than or equal to
=	equal to
>	greater than
>=	greater than or equal to
<> or #	not equal to
$	contained in

When character strings are compared, they are evaluated by the position of each character in the standard ASCII table. The first character in one string is compared to the first character in the other string. If the two are equal, the second characters are compared and so on.

Character strings are identified with quotation marks. If the text input is being compared to a string within a variable, the STRING operation must be done on one of the variables.

Example: STRING(@@) = @B

Logical Operators

You can specify an expression that uses one or more logical operators (.AND., .OR., and .NOT.) to compare expressions that contain numbers, character strings, and the logical constants .T. and .F.

.AND. asserts that both expressions are true.
.OR. asserts that at least one of the comparisons is true.
.NOT. asserts that the negative of the comparison is true.

Related Topics:

Using the Text Input Dialog Box

& operator

The character operator "&" concatenates (adds) one string to another.

Example: @@ = ("Hello " & "George")

In this example a user is asked to enter "Hello George". This expression tests whether the input is equal to the string "Hello George".

Related Topics:

Text Validation Rules

Extract(start,len,str) operator

The character operator EXTRACT() extracts a portion of a string. The syntax for this operator is EXTRACT(start,len,str), where:

start is the position of the first character you want to extract
len is the length of the portion you want to extract
str is the name of the variable being manipulated

Example: EXTRACT(1,3,@@) = "603" where @@ = "603-555-1234"

In this example, a user is asked to enter a telephone number (with a 603 area code). The expression extracts the first three characters of the input and tests whether they are equal to the string "603". To extract "603" for comparison, specify 1 which is the position of the "6", specify 3 which is the number of characters being extracted, and specify @@ which is the input variable being acted upon.

Related Topics:

Text Validation Rules

Upper(str) and Lower(str) operators

The character operator UPPER() converts a string specified within the parentheses to uppercase. The character operator LOWER() converts the string to lowercase. If the Text Format is set to Standard the Text Input dialog box, the user enters text input as it appears on the screen and is stored. For example, if the user types "Red", the value "Red" is stored in the input variable. Use UPPER() or LOWER() to convert the value in the input variable to upper or lower case when it is compared to a string for validation.

Example: UPPER(@@) = "RED"

This expression converts the input string to uppercase and compares it to "RED". It stores it as "Red", which the user typed.

When you use a case conversion operator it converts the input string for the validation, but does not permanently change it.

Related Topics:

Text Validation Rules

Numeric Validation Rules

When you choose Options>> from the Numeric Input dialog box you can create an expression in the validation area. The expression you create evaluates the user's input to determine whether it is acceptable. As an example, consider the user response to: "Enter a number between 1 and 100." The expression in the validation view box for this case is:

@@ >= 1 .AND. @@ <= 100

The user input is validated according to this expression. If it is greater than or equal to 1 and less than or equal to 100, it is accepted. If it is some other value, the system beeps and the user must re-enter input.

After you enter all of the input validation information, choose OK. The validation area is removed. Choose OK again to close the remaining portion of the Numeric Input dialog box.

Numeric Operators

In the validation view box, you can specify an expression that uses one or more numeric operators.

The following table describes the operations that are supported.

Operator	Meaning	Example
+	addition	2.3 + 2 = 4.3
-	subtraction	6 - 1 = 5
*	multiplication	4 * 3 = 12
/	division	12 / 4 = 3
** or ^	exponent	2 ^ 3 = 8
SQRT(x)	square root	SQRT(4) = 2
ABS(x)	absolute value	ABS(-52) = 52
INT(x)	integer value	INT(2.3) = 2
		INT(2.8) = 2
		INT(2.854) = 2
ROUND(x)	round off value	ROUND(2.3) = 2
		ROUND(2.8) = 3
		ROUND(2.3333) = 2
SIN(x)	sine value	SINE(30) = 0.5
COS(x)	cosine value	COS(60) = 0.5
TAN(x)	tangent value	TAN(0) = 0.0
LOG(x)	natural logarithm	LOG(1.0) = 0.0
LOG10(x)	logarithm base 10	LOG10(10) = 1.0
+	positive	+2.3 = 2.3
-	negative	-3 = -3

You can use these numeric operations individually, or you can combine them. Normal mathematical and left-to-right precedence rules apply, unless parentheses are used. For example, while 2 * 3 + 4 = 10, whereas 2 * (3 + 4) = 14.

Relational Operators and Numbers

You can specify an expression that uses a relational operator, such as the = sign, to compare numbers.

Operator	Meaning	Examples
<	less than	@@ < 4
<=	less than or equal to	@@ <= 1
=	equal to	@@ * 2 = 4
>	greater than	@@ > 4
>=	greater than or equal to	@@ >= 3
<> or #	not equal to	@@ * 3 <> 4

Logical Operators

You can specify a validation expression that uses one or more logical operators (.AND., .OR., and .NOT.). These expressions can contain numbers, character strings, and the logical constants .T. and .F.

.AND. asserts that both expressions are true.

.OR. asserts that at least one of the comparisons is true.

.NOT. asserts that the negative of the comparison is true.

Related Topics:

Using the Numeric Input Dialog Box

Using the Mouse Input Dialog Box

When you select Mouse and choose Options... from the Input Selection Editor dialog box, the Mouse Input dialog box appears and contains the following components:

Result box
Mouse Format area
Mouse Button area
Cursor Shape text box
Input Validation check box
Input Area text box

Mouse Format area

Specify the action the user must perform to use the mouse to select a point.

Mouse Down - Pressing the mouse button down (clicking) selects a point. (This is the default.)

Mouse Up - Releasing the mouse button selects a point.

Double Click - Pressing the mouse button down twice in quick succession selects a point. (The double-click rate is set in the Windows Control Panel.)

Related Topics:

Using the Mouse Input Dialog Box

Mouse Button area

Specify which mouse button is active.

Left - Only the left mouse button is active. (This is the default.)

Right - Only the right mouse button is active.

All - All mouse buttons are active.

Related Topics:

Using the Mouse Input Dialog Box

Cursor Shape

Specify the cursor shape. Your options are:

Arrow - Standard arrow cursor. (This is the default.)
Indexed Hand - Closed hand with index finger pointed upward.
Left Arrow - Arrow cursor pointing left.
Right Arrow - Arrow cursor pointing right.
Up Arrow - Arrow cursor pointing up.
Down Arrow - Arrow cursor pointing down.
Crosshair - Large plus sign
I Beam - Text I-beam cursor.
None - A cursor is not displayed. Use this option for touch input.

Related Topics:
Using the Mouse Input Dialog Box

Input Validation check box

The Input Validation option lets you select an active area of the screen where the mouse action can occur. If Input Validation is selected, the selected cursor shape only occurs in the active area. When the mouse cursor is moved within the non-active areas, the default arrow cursor displays.

If the Input Validation option is not selected, the Input Area box is greyed out. When you select Input Validation, you can enter data in the Input Area box to define an area on the screen where a user is permitted to select a point. Anywhere outside of this area, the mouse cursor appears as an arrow shape. If the user tries to select a point outside the valid area, the system beeps.

Related Topics:

Using the Mouse Input Dialog Box

Input Area text box

The data you enter in the Input Area box consists of four numbers. The first two numbers are the x,y coordinates of the upper left corner of the area. The second two numbers are the width and height of the area. The Area Editor is available from the drop-down list box of the Input Area box to allow you to visually select an area of the screen. When you close the editor, the data that defines the area you selected is returned to the Input Area box.

Related Topics:

Using the Mouse Input Dialog Box

Using the Numeric Input Dialog Box

When you select Numeric and choose Options... from the Input Selection Editor dialog box, the Numeric Input dialog box appears and contains the following components:

Result box
Initialize Input Variable check box
String Variable Name
Display area
Format area
Input Termination area
Options >> button

String Variable Name

A user's numeric input is stored in the Input Variable. Optionally, use the String Variable Name text box to specify the variable in which you want to store the string value of the numeric input.

The corresponding drop-down list box lets you access the Variable Selector. Use the Variable Selector to select a variable.

When a string value is saved to this variable, commas and decimal points that are part of the picture are also saved. The string that you save can be useful, for example, if you want to redisplay the number at some later time, with the proper punctuation.

Related Topics:

Using the Numeric Input Dialog Box

Format area

Type options

Specify the kind of numbers that the user can enter from the keyboard.

Real - The user can enter a real number. If Real is selected, the Number of Decimal Places box is active. The acceptable range for real numbers is 0.0000000000 to 99,999,999,999.9999999999

Integer - The user can enter an integer. The acceptable range for integers is -99,999,999,999 to +99,999,999,999

Decimal Places

If Real is selected as the Number Format, specify the number of positions you want to be to the right of the decimal point. For example, if the input is intended to be numbers such as 143.24 or 1.06, specify 2 for the Number of Decimal Places. The acceptable range of values for the Number of Decimal Places is 0 to 10.

Maximum Length

Specify the length of the input the user can provide. The acceptable range for input length is 1 to 22 for real numbers, and 1 to 11 for integers. When you are specifying a length for real numbers, include the decimal point position in your count.

Picture

Predefine where the commas and decimal point appear in numeric input. Pictures consist of 9s, Zs, commas, and a decimal point. The length of the picture (including punctuation) must match the Input Length value and the position of the decimal point must match the Number of Decimal Places value.

9 - Means the user must provide a digit for this position. For example, if a picture is defined as 999.99 the user must type 5 digits. The user can type a 0 in any of these positions. If the user types a 0 in one or more leading or trailing positions, such as the 034.20, the 0s are saved as part of the numeric value. (This means the number stored in the input variable is 034.20)

Z - Means the user can optionally provide a digit for this position. For example, if a picture is defined as ZZZ.ZZ the user can type from 0 to 5 digits. The user can type a 0 in any of these positions. If the user types a 0 in one or more leading or trailing positions, such as 034.20, the 0s are not saved as part of the numeric value. (This means the number stored in the input variable is 34.20)

, (comma) - Means a comma appears on the screen as part of the display. However, the comma is a special character that is not saved as part of the numeric value stored in the input variable.

. (decimal point) - Marks the position of the decimal point.

Note: If a picture uses both Zs and 9s. Zs can occupy only leading or trailing positions. Z99.9Z is valid. Z9Z.99 is not valid. (The Z, which represents an optional digit position cannot occur in the middle of the number.)

Related Topics:

Using the Numeric Input Dialog Box

Using the Date Input Dialog Box

When you select Date and choose Options... from the Input Selection Editor dialog box, the Date Input dialog box appears and contains the following components:

Result box
Initialize Input Variable check box
String Variable Name
Display area
Format area
Input Termination area
Options >> button

String Variable Name

There are two ways to store a date. You can store it in one form in the variable you specified in the Input Variable text box of the Input icon. You can store it in another form in a variable you specify in the String Variable Name text box of the Date Input dialog box.

In the Input Variable text box, the entered date is always treated as a character string and is automatically converted to database format. (Database format is CCYYMMDD. An example of a database formatted date is 19910123, which is January 23, 1991.)

In the String Variable Name text box you can store the date exactly as it appears on the screen. This way, when the user enters a date such as Jan 23, 1991, the date is stored to the string variable @DATE in the form "Jan 23, 1991".

The corresponding drop-down list box for String Variable Name lets you access the Variable Selector. Use the Variable Selector to select a variable. When you exit the Variable Selector, the selected variable is returned to the String Variable Name text box.

Related Topics:

Using the Date Input Dialog Box

Format area

Date

Specify the order in which the user can enter date information from the keyboard. Select one of the following options:

Month Day Year (This is the default.)
Day Month Year
Year Month Day

Month

Specify the format for the month portion of the date.

MM - The month is represented by two digits (01, 02, etc.). (This is the default.)

MMM - The month is represented by three characters (Jan, Feb, etc.).

Year

Specify the format for the year portion of the date.

YY - The year is represented by two digits (91,92,etc.). (This is the default.)

CCYY - The year is represented by four digits (1991,1992, etc.).

Separator

Specify the symbol used to separate one part of the date from the next. These symbols are not entered by the user. They are automatically displayed on the screen during input. The following options are available:

/	Example: 12/22/91 (This is the default.)
.	Example: 12.22.90
,	Example: Jan 12, 1991
-	Example: 03-22-92
a space	Example: Jan 04 1991

Note: The comma date separator only applies when MMM and YY are selected or when CCYY is selected. The comma is only displayed after the day element.

Date Validation Rules

When you choose Options>> from the Date Input dialog box, you can create an expression in the validation area that evaluates the user's input to determine whether it is acceptable. As an example, consider response to: "On what date did a human first walk on the moon?" The answer is July 20, 1969; therefore, the following expression is included in the validation view box:

@@ = "19690720"

When the user types a response, the input is validated according to this expression. If the input is 19690720, it is accepted. If it is any other date, the system beeps and the user must re-enter input. Remember, the value stored in the input variable is automatically converted to database format. This means that in a validation expression (as in the previous example), @@ must be compared to dates in database format.

After you enter all of the input validation information, choose OK. The validation area is removed. Choose OK again to close the remaining portion of the Numeric Input dialog box.

The EXTRACT() Operator and Dates

The character operator EXTRACT() extracts a portion of a string. The syntax for this operator is EXTRACT(start,len,str), where:

start is the position of the first character you want to extract
len is the length of the portion you want to extract
str is the name of the variable being manipulated

The STRING() Function and Dates

Dates are stored and evaluated as character strings. Therefore the dates that they are compared to must be strings that are identified with quotation marks. If the date you are comparing input to is represented by a variable, the STRING operation must be done on one of the variables.

Example: STRING(@@) = @DATE

Relational Operators and Dates

When character strings are compared, they are evaluated by the position of each character in the standard ASCII table. The first character in one string is compared to the first character in the other string. If the two are equal, the second characters are compared and so on. (See Appendix B for an ASCII conversion table.)

Example: @@ = "19910215" evaluates to true if @@ contains "19910215".

Example: @@ = "19910215" evaluates to false if @@ contains "19810215", because the first two characters are equal and the third characters "9" and "8" are not equal in the ASCII table.

Operator	Meaning	Examples
<	less than	@@ < "19910214"
<=	less than or equal to	@@ <= "19910214"
=	equal to	@@ = "19910214"
>	greater than	@@ > "19910214"
>=	greater than or equal to	@@ >= "19910214"
<> or #	not equal to	@@ <> "19910214"

Logical Operators

You can specify a validation expression that uses one or more logical operators (.AND., .OR., and .NOT.).

These expressions can contain numbers, character strings, and the logical constants .T. and .F.

.AND. asserts that both expressions are true.

.OR. asserts that at least one of the comparisons is true.

.NOT. asserts that the negative of the comparison is true.

Example: A user is asked to enter a date between Jan 15, 1991 and Jan 15, 1992. The validation expression is as follows:

@@ < "19920115" .AND. @@ > "19910115"

InputMenu Icon

The InputMenu icon is used to get input from the user. Specifically, it lets the user choose from one of two or more options on the screen. These options are usually menu choices or multiple choice answers to a question.

Important: Do not use the InputMenu icon if you are using live objects (available through the SmartObject Editor) in your application. The SmartObject Editor lets you use interactive objects to create menus.

Although the InputMenu icon is a separate icon in the library and can be built into other types of structures; it is frequently used as part of the Menu composite. In the Menu composite, a Display icon typically precedes an InputMenu icon. The Display icon displays the menu (a graphic file or a SmartObject file) on the screen. In addition to selection areas, the screen may provide directions such as "Click in an area to select an activity."

The purpose of the InputMenu icon is to take the image on the screen, and make portions of it selectable. The icon below the InputMenu icon is not executed until either the user makes a response or a specific time-out period expires.

When you add content to an InputMenu icon, you define which areas on the screen you want to be selectable, and the action the user must take in order to select an area. When you define selectable areas, each area is assigned a number according to the order in which it is defined.

When a user responds to an InputMenu icon by an action such as clicking on or touching an area of the screen, the number associated with the area the user selects is automatically stored in the system variable @_SELECTION.

As in the Menu composite, an InputMenu icon is frequently followed by a Branches composite. The Branches composite contains several If icons, each of which compares the value in @_SELECTION to a number such as 1, 2, 3, etc. If @_SELECTION = 1, the first If icon tests true and the branch below is executed. If @_SELECTION = 2, the second If icon tests true and the branch below is executed, and so on.

In addition to @_SELECTION, the InputMenu icon uses @_USERTIME, @_TIMEOUT, and optionally, @_TEXT_AREAS.

Content Editor Text Boxes:

Selection Areas
Time Limit
Selection Indicator
Feedback
Selection Labels

@_TEXT_AREAS

In the InputMenu icon, you define the selectable areas in a screen display in the Selection Areas text box. When you display a graphic file you typically use the Input Template Editor (through the InputMenu icon) to identify selectable areas. This editor returns data that describes size and location of each selectable area to the Selection Areas text box. You can use this same method to identify selectable areas in a page of a SmartObject file, or, if you already identified certain areas (or objects) in a page as selectable within the SmartObject Editor, you can specify @_TEXT_AREAS in the Selection Areas text box.

When you create a SmartObject page, text, graphic and OLE objects that are not live can be designated it as selection areas. Also within the SmartObject Editor, you identify the number associated with each selection area.

When a Display icon displays a SmartObject page with selection areas, the coordinates of these areas are automatically placed in @_TEXT_AREAS. Therefore, when you follow the Display icon with an InputMenu icon, you can include @_TEXT_AREAS in the Selection Areas text box, to directly reference the selectable areas in the SmartObject page.

Selection Areas

Specify the areas on the screen that you want to be selectable. The order in which the areas are listed in the Selection Areas text box is important because each area is assigned a number (1,2,3,etc.) according to this order.

Acceptable values are: one or more groups of 4 numbers, **@_TEXT_AREAS**, or a variable.

It is likely that you will use the Input Template Editor to automatically generate groups of numbers each of which defines a rectangular, selectable area on the screen. The numbers within a group are separated by commas, the groups of numbers are separated by semicolons. The first two numbers in every group are the x,y coordinates of a rectangular area. The second two numbers in every group are the width and height of a rectangular area.

Drop-down List Box Items:

Input Template Editor
Variable Selector

Time Limit

Specify whether or not there is a limit to the amount of time a user can take to respond to the input screen. If you do not want a time limit, specify 0. If you do want a time limit, enter a numeric value.

If you do enter a time limit and the user does not provide input within the specified time, the icon below the InputMenu icon is executed. When the InputMenu icon is executed, two system variables are set. @_TIMEOUT is set to 1 if timeout occurred before the user responded. @_TIMEOUT is set to 0 if the user responded before timeout occurred. Also, the system variable @_USERTIME contains the number of 100ths of a second it took the user to respond.

Acceptable values are: a positive real number (such as 4 or 4.3) or a variable.

Drop-down List Box Items:

Variable Selector

Selection Indicator

Specify the kind of action the user must take in order to select an area.

Although each option in the drop-down list box gives the user the ability to make a selection by a different means, such as clicking a mouse, or touching the screen, each option also lets the user make a selection by partially or exclusively using the keyboard. The arrow keys are always active for an input screen and you can designate a key or multi-key combination for each selectable area. To designate keyboard actions, include a key or multi-key combination in the Selection Labels text box for each selectable area.

Acceptable values are: **move**, **press**, **return**, **touch**, or a variable.

Drop-down List Box Items:

move - The user can make a selection in two ways: 1) move the mouse pointer into an area, or 2) press a key or multi-key combination. The different key or multi-key combination the user can press for each area is designated in the Selection Labels text box.

press - The user can make a selection in two ways: 1) click the mouse button on an area, or 2) press a key or a multi-key combination. The different key or multi-key combination the user can press for each area is designated in the Selection Labels text box.

return - The user can make a selection in two ways: 1) click the mouse button on an area and press RETURN, or 2) press a key or a multi-key combination and press RETURN. The different key or multi-key combination the user can press for each area is designated in the Selection Labels text box.

touch - The user can make a selection in two ways: 1) touch the screen, or 2) press a key. The different key or multi-key combination the user can press for each area is designated in the Selection Labels text box.

Variable Selector

Feedback

Specify a visual effect that you want to be part of the menu. For example, when you specify press for a selection indicator, and block flash for feedback, when the menu appears, the first area (associated with the number 1) is highlighted automatically. As the user cycles through the selection areas, by moving the mouse pointer or using the arrow keys, the current selection area is highlighted. When the user actually clicks or presses the designated key or multi-key combination, the current highlighted area flashes, and the InputMenu icon is executed.

Acceptable values are: **block**, **block flash**, **box**, **box flash**, **none**, or a variable

Drop-down List Box Items:

block - When the menu appears the first area is automatically highlighted. If a user changes focus to a different area (via the mouse or arrow keys), the new area is highlighted instead.

block flash - When the menu appears the first area is highlighted automatically. If a user changes focus to a different area (via the mouse or arrow keys), the new area is highlighted instead. When the user actually selects an area by clicking, pressing a designated key, etc., the selected highlighted area flashes three times.

box - When the menu appears the first area is automatically enclosed in a box. If a user changes focus to a different area (via the mouse or arrow keys), the new area is enclosed in a box instead.

box flash - When the menu appears the first area is automatically enclosed in a box. If a user changes focus to a different area (via the mouse or arrow keys), the new area is enclosed in a box instead. When the user actually selects an area by clicking, pressing a designated key, etc., the box surrounding the selected area flashes three times.

none - No visual effect occurs.

Variable Selector

Selection Labels

Although each selection indicator option gives the user the ability to make a selection by a different means, such as a clicking a mouse, or touching the screen, each option also lets the user make a selection by partially or exclusively using the keyboard.

Each selectable area has multiple keys associated with it by default. For example, if you do not specify any selection labels, when a user presses the F1 key, the 1 key, the keypad 1 key, or the A key, the first area is selected. When the user presses the F2 key, the 2 key, the keypad 2 key, or the B key, the second area is selected, and so on.

If you do not want to use these default keys, you can optionally, use the Selection Labels text box to designate a different key or multi-key combination for each selectable area. The first key you specify corresponds to the first area, the second key corresponds to the second area, etc. Use a comma to separate each key label from the next.

Note: When you specify any keys or multi-key combinations in the Selection Labels text box, the default keys for the selection areas are disabled.

The following keys are available as single key selection labels:

A through Z keys	HOME (Home key)
0 through 9 keys	INS (Insert key)
0 through 9 (keypad)	TAB (Tab key)
BS (Back Space key)	NEXT (Page Down key)
DEL (Delete key)	PRIOR (Page Up key)
END (End key)	LEFT (left arrow key)
RETURN (Enter key)	RIGHT (right arrow key)
ESC (Esc Key)*	UP (up arrow key)
F1... F2 (F1 through F12)	DOWN (down arrow key)
	PAUSE (pause/break key)

* The Esc key is used in the IconAuthor Authoring system to escape from a running application and return to the IconAuthor window. However, under some circumstances, you can still use the Esc key in your applications because Esc is interpreted as a standard non-displayable character by the IconAuthor Presentation system. The only case in which Esc is not available for use is if it is the designated break key for the Presentation system. To determine which key or multi-key combination is reserved as the break key for the Presentation system, look at the BreakKey= entry in the IconAuthor section of your PRESENT.INI file.

To specify a multi-key combination for a selectable area use one or any combination of SHIFT, CONTROL, or ALT followed by one of the valid single keys described previously. Use a hyphen to separate one key from another if they are all part of a multi-key combination. For example, CONTROL-SHIFT-F7, or ALT-F4 are valid selection labels.

Acceptable values are: an allowable single key, a multi-key combination, or a variable.

Drop-down List Box Items:

Variable Selector

Line Icon

The Line icon dynamically draws a line.

Use the text boxes in the Line icon Content Editor to specify the endpoints and the width of the line.

By default the line color is black (the default outline color). To create a line using an alternative color, precede the Line icon with a Color icon. The outline color you specify is in effect until you use another Color icon.

Suggested Uses:

- Use a line to underline or point to an item or text on the screen
- Use a line to connect one item to another on the screen

Content Editor Text Boxes:

Start Point
End Point
Line Width

Start Point

In order to begin defining the line, specify the coordinates of the start point.

Acceptable values are: a pair of screen coordinates or a variable.

Drop-down List Box Items:

Location Editor
Variable Selector

End Point

To finish defining the line, you specify the coordinates of the end point.

Acceptable values are: a pair of screen coordinates or a variable.

Drop-down List Box Items:

Location Editor
Variable Selector

Line Width

Specify the width (in pixels) of the line. Make sure to use a line width of 1 pixel or more.

Acceptable values are: any positive integer (greater than 0) or a variable.

Drop-down List Box Items:

assorted numbers - Frequently used line widths.

Variable Selector

LoadVar Icon

The LoadVar icon lets you load several variables into memory from one variable file.

Note: You can also use a Variable object (available through the SmartObject Editor) to load variables.

A variable file contains one or more single value or indexed application variables with or without values. You can create a variable file by creating an ASCII file with a text editor such as Notepad. Or you can create a variable file by saving variables in an application to a file (using the SaveVar icon).

When you use a LoadVar icon and specify the name of a variable file, you load all the variables in the file into memory at once. This is more convenient than using a Variable icon to define and/or assign a value to one variable at a time.

Suggested Uses:

- Load a variable file that contains several single value and indexed variables
- Load a variable file at the beginning of the main application to reset the values of certain variables. (Unless a variable is initialized in the course of the application it may contain an unwanted value the next time the application is run.)

Content Editor Text Boxes:

Filename
Control
Variable Name

Related Topic:

Using Variables

Filename

Specify the name of the variable file that contains one or more variables you want to load into memory.

Acceptable values are: a variable filename or a variable.

Drop-down List Box Items:

Directory
Variable Selector

Control

Indicate whether you want to load all or one of the variables in a variable file.

Acceptable values are: **all**, **single**, or a variable.

Drop-down List Box Items:

all - Loads all variables in the specified file .

single - Loads the variables specified in the Variable Name text box. If an array name is specified, the entire array is loaded into memory.

Variable Selector

Variable Name

If you want to load one variable or an array from a variable file (you entered "single" in the Control text box), specify the name of the variable.

Acceptable values are: a variable or an array name.

Drop-down List Box Items:

Variable Selector

Loop Icon

The Loop icon is a composite that causes the application to repeatedly execute a particular group of icons. A loop can use the *same* logic over and over again so you don't have to build the same logic into your structure repeatedly.

Note: The composite Loop icon is different from the composite LoopIndex icon. Use a composite Loop icon in your structure where you want the *user* to control how many times a group of icons is executed.

The Loop icon is made up of three icons: the Loop icon, the LoopStart icon, and the LoopEnd icon. These icons provide the framework for the repetitive part of your structure.

Include the icons that you want to execute repeatedly, between the LoopStart and the LoopEnd icons. Only the lead icon in this composite, named Loop, has a Content Editor which allows you to change the name of the composite.

Related Topics:

Exiting from Loop Composite
Loop Composite versus LoopIndex Composite

Exiting from a Loop Composite

You must include an Exit icon with the icons you build between the LoopStart and LoopEnd icons. Otherwise, the icons in the loop will repeat infinitely. The Exit icon must specify either "loop", a positive integer that indicates a number of nested loops, or "application" in the Exit From text box.

If you specify "loop" in the Exit From text box of the Exit icon, the loop is exited and execution flows to the icon below the lead icon in the Loop composite. If you specify a number, that number of nested loops are exited, and execution flows to the icon below the lead icon in the outer Loop composite. For example, if you specify "2" in the Exit From text box of the Exit icon, the loop and the loop in which it is nested are exited. If you specify "application" in the Exit From text box of the Exit icon, this causes an exit from the loop *and* the main application.

Related Topics:

Loop Icon
Exit Icon

Loop Composite versus LoopIndex Composite

The composite Loop icon is different from the composite LoopIndex icon.

The composite Loop icon repeats a variable number of times and is used when you want the *user* to control how long the loop should continue. When users want to exit the loop (are finished with the activity), they choose the path that contains the exit from the loop.

The composite LoopIndex icon repeats a fixed number of times and is used when *you* want to control the number of times a group of icons repeats.

Related Topics:

Loop Icon
LoopIndex Icon

LoopIndex Icon

The LoopIndex icon is a composite that causes the application to repeatedly execute a particular group of icons. A loop can use the same portion of the structure over and over again, so you don't have to build it into your application repeatedly.

Note: The composite LoopIndex icon is different from the composite Loop icon. Use a composite LoopIndex icon in your structure when *you* want to control the number of times a group of icons repeats.

The LoopIndex icon is made up of three icons: the LoopIndex icon, the LoopStart icon, and the LoopEnd icon. These icons provide the framework for the part of your structure you want to repeat.

To use the LoopIndex icon, you include the icons that you want to execute repeatedly, between the LoopStart and the LoopEnd icons. In the LoopStart icon Content Editor you indicate how many times you want the loop to execute.

The loop is exited when it has completed the number of specified repetitions, or when it reaches an Exit icon. The Exit icon is not required, however, it is used to give users an opportunity to exit from the loop if they want.

Only the LoopStart icon requires that you enter values in the Content Editor.

LoopStart Icon

The LoopStart icon is the only icon in the LoopIndex composite that has a Content Editor that requires values.

LoopStart Icon Content Editor Text Boxes:

Index Variable
Number of Loops
Initialization

Index Variable

Specify the variable that will keep track of the number of times the loop has repeated. This variable is called a "counter". You may want to know how many times the loop has repeated. You may store this information in a variable such as @COUNT, and use @COUNT elsewhere in the application.

Drop-down List Box Items:

Variable Selector

Number of Loops

Specify the number of times you want the loop to repeat.

Acceptable values are: a positive integer (greater than 0) or a variable.

Drop-down List Box Items:

Variable Selector

Initialization

Specify whether you want the variable (in the Index Variable text box) to be initialized (set to 1) the first time the loop is executed.

Acceptable values are: **no**, **yes**, or a variable.

Drop-down List Box Items:

no - the variable in the Index Variable text box is not initialized and starts at its current value. If a variable was not initialized at some other point in the application, it could have an ever-increasing value each time the application is run. Eventually, if the variable is equal to or greater than the value in the Number of Loops value, the loop will not execute.

yes - The variable in the Index Variable text box is initialized to 1.

Variable Selector

MCI Icon

The MCI icon provides direct access to MCI (Media Control Interface) which is part of the Multimedia Extensions software.

Use MCI to interact with the following multimedia elements: CD-Audio, digital audio, MIDI, and animation. Each MCI icon executes an MCI command and gets a return code (if any) as well as an error code and matching error message.

Important: For the quickest and easiest way to take advantage of the multimedia capabilities of MCI, use the SmartObject Editors Audio and Movie objects instead of the MCI icon.

Content Editor Text Boxes:

MCI Command
MCI Result
MCI Error Number
MCI Erroor Message

Related Topics:

CD-Audio Composite
MIDI Composite
WaveAudio Composite

MCI Command

Specify an MCI command.

The MCI command can optionally include one or more IconAuthor variables. A variable can be used to represent the entire command string or it can be used to represent part of a command string. For detailed information on the MCI commands and syntax, open the help file called MCISTRWH.HLP.

Also, you can issue the IconAuthor command "QUERY ICONAUTHOR WINDOW". This command returns the IconAuthor window handle needed by the animation player.

Drop-down List Box Items:

MCI Command Selector
Variable Selector

MCI Result

Specify a variable in which you want to store information returned from MCI. For example, your MCI command string can include a status command to check whether a particular device is ready to be used. The returned values are listed in the MCI reference. The application can take a particular branch based on the value that is returned.

Drop-down List Box Items:

Variable Selector

MCI Error Number

Specify a variable in which you want to store an error number returned from MCI. The error number can be used to see if the end user system is functioning as expected. For example, it can indicate whether the end user has inserted a CD-Audio disc.

For convenience, the default variable @error_number automatically appears in this text box. Use this variable, or specify a different one.

Drop-down List Box Items:

Variable Selector

MCI Error Message

Specify a variable in which you want to store an error message returned from MCI. This text box works similarly to the MCI Error Number text box.

For convenience, the default variable @error_message automatically appears in this text box. Use this variable, or optionally, specify a different one.

Drop-down List Box Items:

Variable Selector

MCI Command Selector

The MCI Command Selector helps you construct a command string to include in an MCI icon. The selector makes the building of command strings easier because it automatically provides the basic syntax for a command you decide to use. The first steps in using the MCI Command Selector require you to choose the desired device and command. The selector then constructs as complete a string as possible given your selections. It identifies, in square brackets ("[ ]"), any parameters within the string that need further definition.

Prerequisites to Using the MCI Command Selector

Because you need to make certain basic decisions, such as which device and command to choose, use of the MCI Command Selector requires that you have basic familiarity with the rules for constructing MCI commands.

Using the MCI Command Selector

To construct a command string

1. From the Device drop-down list, select the device you want to use.

   The devices that appear in the list are those that are registered with MCI.

   Note: If your system has an MCI device that does not strictly adhere to MCI registration conventions, the device may not be available via the MCI Command Selector. In this situation, you will not be able to use the selector for the device, but you will still be able to type a command string directly into the MCI icon Content Editor.

2. From the Command drop-down list, select the command you want to use.

   The list displays the most commonly used commands associated with the selected device. If a command is not available via the selector, you can still use it by typing a command string directly into the MCI icon Content Editor.

   As soon as you choose a command, two changes occur in the selector. First, the items in the Command List change to display the most common variations for the selected command. Second, the MCI Command text box at the top of the dialog box displays the correct syntax given the selected device, the selected command, and the topmost variation of the chosen command.

3. As necessary, use the Command List to choose a different variation of the selected command.

   The MCI Command text box changes again to reflect a different selection from the Command List. When you finish selecting information from the three list boxes, the MCI Command text box displays as complete a string as possible given your selections. Any parameters that appear within square brackets ("[ ]") need further definition.

4. Complete the command string in the MCI Command text box by providing values for all the bracketed parameters.

   [path/file] - To specify a path and file for a command string, click on the Browse... button. A Browser dialog box appears. Select the desired file, click OK and the chosen path and filename description automatically replaces the [path/file] expression in the command string.

   [device_name] - To specify a device name for a command string, double-click on [device_name] in the MCI Command box (to highlight the expression) and type the name of the device. Note that for an open command, the selector automatically includes the device name for you.

   [alias_name] - To specify an alias for a device name, double-click on [alias_name] in the MCI Command box (to highlight the expression) and type the alias. (Use this alias in subsequent command strings that refer to the same device.)

   [position] - To specify a "from" or "to" position, double-click on [position] in the MCI Command box (to highlight the expression) and type the position.

   When you have replaced all the bracketed parameters with the required values.

5. Click on OK.

   The selector closes and the command string is automatically returned to the MCI icon Content Editor.

Important: The MCI Command Selector does not do a validation check on a string before returning it to the Content Editor. This means that if your string contains one or more errors, they will not be detected. IconAuthor will only detect an error when you attempt to execute the icon.

Menu Icon

The Menu icon is a composite that makes it easier for you to create a part of an application that presents the user with a menu screen. The menu provides choices of several activities or options from the menu screen. The Menu composite is a loop structure. When the user makes a choice from the menu, a branch is taken, an action occurs, and then the loop begins again (the menu is redisplayed). This process continues until the user exits the loop (until an Exit icon is reached).

Important: If your application uses live objects (available through the SmartObject Editor) do not use the Menu composite. Use the ObjMenu composite instead.

To use the Menu composite, you must add information to at least some of the Content Editors. In some cases you may not need to add information to a Content Editor, but you still need to open the Content Editor and choose OK to accept the current text box values.

The Menu composite is a framework. You need to add other icons to the structure to make it perform as required. For example, by default, there are four If icons, labeled **1**, **2**, **3**, and **4** in the Menu composite. Your application may need a smaller or larger number of branches. You can cut, copy, paste, and edit the content of If icons to create the required number of branches.

Also, you must add the icons that cause a particular event to occur when a branch is taken to this framework. Under each If icon, you add a series of icons to fulfill the user's selection. For example, if the first choice is selected from a menu, labeled "Tutorial", the icons in the first branch must present a tutorial.

Suggested Uses:

- Create a menu or nested menu within your application
- Present data with user selection buttons for advancing or displaying previous information (page-turning)

The following icons make up the Menu composite:

Menu Icon - Marks the beginning of the Menu composite. The Content Editor contains only a Composite Name text box to let you customize the name of the composite.

LoopStart Icon - Marks the beginning of the loop within the Menu composite. Does not have a Content Editor.

Display Icon - Displays a graphic file or SmartObject file that is the menu screen.

Input Menu - Lets you make areas on the screen selectable. These areas correspond to the choices in the previously displayed graphic or SmartObject file. You can specify how the user responds to the screen (clicking, pressing RETURN, etc.). The value associated with the selection made is stored in the system variable @_SELECTION. For example, if the user clicks on the first area you made input selectable, @_SELECTION = 1, if the user clicks on the second area @_SELECTION = 2, and so on.

Choices Icon - Marks the beginning of the Choices composite within the Menu composite. The Choices composite is similar to the composite Branches Icon because it causes the application to take a particular branch or path of execution depending upon the selection made by the user. The Choices composite is only different from the Branches composite because the Content Editor of each If icon (labeled 1 through 4) at the top of each branch already contains values.

1,2,3, and 4 Icons - These are the four If icons built into the Menu composite. Each of these icons marks the beginning of a different branch. After the user chooses a selectable area on the screen, the 1 icon compares the value of @_SELECTION to the number 1. If @_SELECTION = 1, the condition is true, and execution flows downward to any icons built into the first path. If @_SELECTION $\neq$ 1, execution flows to the right to the 2 icon. Each of the numbered If icons makes a similar comparison until one of the conditions is true.

If the 1 icon, 2 icon, or 3 icon test true, the corresponding branch is executed. After the branch completes, the loop begins again and the menu is redisplayed. If the 4 icon is the one that tests true, the corresponding branch (the Exit icon) is executed.

Exit Icon - This is the only icon in the fourth branch of the Menu composite. It allows the user to exit from the menu composite. (Without an explicit exit, the Menu composite loop, like the composite Loop icon, would execute infinitely.) "Loop" is specified in the Exit From text box. When this icon is executed, execution flows out of the loop to the icon below the lead icon in the Menu composite (labeled Menu).

LoopEnd Icon - This marks the end of the loop within the Menu composite. This icon is executed if one of the first three branches (icon **1**, icon **2**, or icon **3**) is executed. After the LoopEnd icon, execution returns to the beginning of the loop within the Menu composite.

MIDI Composite

The MIDI icon is a composite that allows you to play Musical Instrument Digital Interface (MIDI) files if you are using the Multimedia Extensions software and an audio card such as the SoundBlaster (Creative Labs) or the Pro Audio Spectrum (Media Vision).

Important: For the quickest and easiest way to take advantage of the multimedia capabilities of MCI, use the SmartObject Editors Audio and Movie objects instead of the MCI icon.

Three MCI icons form the backbone of the MIDI composite. Although you must be familiar with the MCI syntax in order to fully take advantage of the MCI feature, the MIDI composite already contains some values (that require minimal editing) so that you can quickly start playing MIDI files as part of your IconAuthor applications. Once you become familiar with the MCI syntax you can customize and vary the commands. For information on the MCI command syntax open the help file called MCISTRWH.HLP.

The composite contains the a mini-structure of icons:

1. MCI icon: Contains the command **open c:\iauthor\audio\filename.mid type sequencer alias midi**, where: **open** initializes the device and **c:\iauthor\audio\filename.mid** represents the path and filename of the MIDI file to be played. You must change this information so that it indicates the specific path and filename you are playing. The **type sequencer** parameter is the type of device and **alias midi** specifies the name "midi" as an alternate name for the sequencer device type.
2. MCI icon: contains the command **play midi** which starts the MIDI file playing.
3. Input icon: Causes execution flow to stop at this point and wait for the user to provide input. This icon specifies that the entire screen is input selectable. That means that the user can click anywhere or press any key to cause execution flow to continue.
4. MCI icon: When the user clicks, execution flows to the third MCI icon which contains the command **close midi** to suspend playback and relinquish access to the device.

Hint: If you want to use this composite to play a MIDI file while some other activity is occurring, replace the Input icon with one or more alternative icons. For example, if you use a Display icon (in place of the Input icon) to run an animation script, the audio will play, the animation will run, and when the animation completes, the audio file will be closed.

Content Editor Text Box:

The lead icon in the MIDI composite is labeled "MIDI" and contains only one text box "Composite Name". Enter a different name in this text box to customize the name of the composite.

Related Topics:

MCI Icon
Input Icon

Module Icon

The Module icon is an empty composite icon that you can use to help design and organize your application. Each Module icon used in an application can label a separate major part of the structure. This approach, called *top down authoring,* uses Module icons to build an outline of the planned structure. Once the Modules have been built into an outline of the application, structures that make up each module are created.

The Module icon is the lead icon in a composite that you create. It has only one text box labeled Composite Name. This text box allows you to customize the name of the module.

As you build an outline out of Module icons, you can rename them to represent the different parts of the application. For example, if part of your application will be a test, you can rename that module "Test". If another part of your application will be a demonstration, you can rename that module "Demo".

When the outline is complete, you build structures to the right of each Module icon, just like you do in other composites. At first, a Module icon appears yellow. As soon as you build icons to the right of it, it appears green. (These color specifications assume you are using the IconAuthor default color scheme.)

Suggested Uses:

- Use Module icons to do top down authoring.
- Use Module icons and hide the icons in each Module composite so that at a glance, others will understand the overall logic behind your application's structure.
- Hide the icons in any module that is finished so that at a glance you can tell if a particular module still needs work.
- Use the Find command from the Edit menu to search for the name of a module to jump to an area of the structure more quickly.

MsgBox Icon

The MsgBox icon lets you display a message box as part of your IconAuthor application. You define key information such as the message content, the kinds of buttons that appear, and a variable in which to store the user's selection. For example, your message box can ask "Do you want to continue?" and present the user with two buttons labeled "Yes" and "No." If the user clicks on the Yes button, the string "Yes" is placed in the variable specified in the Variable Name text box.

Content Editor Text Boxes:

Message
Title
Buttons
Icon
Variable Name

Message

Specify the message you want to display. If you type a particularly long message, the words will wrap onto the next line and the message box will appear taller. Acceptable values are a string or a variable.

Drop-down List Box Items:

Variable Selector

Title

Specify the text you want to appear in the title bar of the message box. Acceptable values are a string or a variable.

Drop-down List Box Items:

Variable Selector

Buttons

Specify a keyword that controls which buttons appear in the message box. Acceptable values are keyword or a variable.

Drop-down List Box Items:

Ok
OkCancel
YesNo
YesNoCancel
RetryCancel
AbortRetryCancel
Variable Selector

Icon

Specify the symbol you want to appear in the message box. As an example, if you specify Stop a Stop sign is displayed within the box along with the message. By Windows convention, each symbol is intended to provide immediate visual information to the user about the nature of the message. For example, the Stop Sign indicates an important message such as a warning, whereas the Information symbol indicates that the message is informational.

Acceptable values are keyword or a variable.

Drop-down List Box Items:

exclamation - Generates a circle with an exclamation point.

information - Generates a circle with an "I."

question - Generates a circle with an question mark.

stop - Generates a circle with a stop sign.

Variable Selector

Variable Name

Specify the name of the variable in which you want to store the user's response to the message box. When the user clicks on a button, the name of that button is assigned to the variable.

Drop-down List Box Items:

Variable Selector

Note Icon

The Note icon is for the author only. Use this icon to document the development of your application and to keep track of comments and ideas while you are authoring. You can have as many Note icons in your application as you want because they do not have any effect on how the application will run.

Note

Type your note in this field.

Acceptable Values:

Any keyboard characters are valid.

Drop-down List Box Item:

Note Editor - Accesses the Note Editor which lets you enter as many lines of text as you need. Click in the text box to type your note. Click OK when you are finished.

ObjDelete Icon

The ObjDelete icon deletes live objects when your application no longer has a need for them. This icon cannot delete static objects, the window object, or the system object. Deleting an object at runtime does not affect the original object in the SmartObject file. However, once the object has been deleted from memory, you need to re-display the file in order to create the object again.

Suggested Uses:

- Create and display a SmartObject page that has a live object. When your application no longer needs the object, use an ObjDelete icon to delete the object.
- Create a SmartObject page that has several objects. Assign one family name to the objects that are required by your application at all times. Assign a different family name to those objects that are not required for the long term. Display the page. When your application no longer needs the objects whose purpose is short term, use an ObjDelete icon to delete the entire family of objects.

Content Editor Text Boxes:

Scope
Name

Scope

Specify the scope of objects that you want to delete. Any objects that fall within the specified scope are deleted permanently *except* those that have their DeleteProtected property set to True.

Acceptable values are: Object, Class, Family, Page, All, or a variable.

Drop-down List Box Items:

Object - Indicates that you want to delete a single object.

Class - Indicates that you want to delete an entire class of objects, for example, all buttons or all OLE objects.

Family - Indicates that you want to a family of objects, for example, all objects that have their FamilyName property set to Color Buttons.

Page - Indicates that you want to delete all the live objects on a particular page.

All - Indicates that you want to delete all live objects.

Variable Selector

Name

Specify the name of the object, class, or family you want to delete. Leave this text box blank if you specified a scope of All. If you are deleting a class of objects, use the drop-down list box to access the Object Class Selector.

Acceptable values are an object name, a class, a family name, a page name, or a variable.

Drop-down List Box Items:

Object Class Selector - Accesses the Object Class Selector which lets you select the name of a class for this text box. Click on the class you want to delete and choose OK. The class is automatically returned to the Name text box.

Object Name Selector

Variable Selector

ObjEvent Icon

Use the ObjEvent icon to control how and when your application waits for events. An event is an *action that is recognized* by your application. Some events are generated by a user (such as a mouse click) and some are generated by an object (such as a Timer count down that reaches 0). If an ObjEvent icon detects that an event has occurred, it stores information about the event in system variables so that subsequent icons in your application can evaluate what has happened.

There are three basic ways to use the ObjEvent icon:

In many situations, you set up the ObjEvent icon so that it waits indefinitely for an event to occur. For example, a Display icon displays a SmartObject page and then an ObjEvent icon is defined to wait for an unlimited amount of time. This gives the user as much time as is needed to respond to the display. This is particularly useful if you consider that without the ObjEvent icon, the page would be displayed and execution would proceed immediately. The user would have little or no opportunity to respond to the display.

You can also use the ObjEvent icon to give the user a specific amount of time to respond to the display. For example, define the icon to give a user 30 seconds to answer a multiple choice question and then choose OK. If the user doesn't respond in time, the ObjEvent icon times out and execution flows down anyway.

In a more complex application, you might define the ObjEvent icon so that it checks to see if an event has occurred. If an event has occurred, it processes the event and execution flows downward. If no event has occurred, execution simply flows downward.

Content Editor Text Boxes:

command
timeout

Events

Many actions can occur with live objects, but only those that are recognized by your application are events. For example, a user has the ability to click on an OK button with the left or right mouse button, but by default, only the left mouse click is recognized as an event.

Every event has a name. For example, one event is called ClickLeft for when a user clicks the left mouse button. When an event occurs, the name of the event is stored in the system variable @_Object_Event. Also, the name of the object involved in the event is stored in the variable @_Object_Name. As an example, if a user left mouse clicks on a button called OkButton, the string ClickLeft is stored in @_Object_Event and the string OkButton is stored in @_Object_Name. If the objects ObjectData property has previously been set, it is stored in @_Object_Data.

Once an event occurs, execution can flow beyond the ObjEvent icon and other icons can evaluate the event (via the values stored in the system variables) and branch accordingly.

Most often, events occur when a user interacts with a live object. For example, an event can occur when a user left mouse clicks on a button or when a user double clicks on an OLE object. In other cases, an event is triggered by a live object all on its own. For example, when a Timer object counts down from 30 to 0 seconds, an event occurs when the count reaches 0.

User-Generated Events

In most situations *you* determine which actions generate events by setting certain properties for each live object. Most objects have properties that have the prefix "NotifyOn" that let you specify whether IconAuthor should be notified if a particular action occurs. For example, a button has two properties called NotifyOnClickLeft and NotifyOnClickRight. By default, these properties are set to True and False, respectively. This means that an event occurs when a user left clicks on a button but an event does not occur when a user right clicks on a button. The text that follows "NotifyOn" in the property name is always the string that is stored in @_Object_Event when an event occurs.

Object-Generated Events

In some situations, an object triggers an event without the help of the user. The best example of this is the Timer object. A Timer object can be set to count up or down, to go off periodically, or to go off at a particular time of day. Whenever a Timer goes off an event occurs. The event always causes the string "Alarm" to be stored in the system variable @_Object_Event and the name of the Timer (its ObjectName) is stored in @_Object_Name.

Related Topic:

Event Queuing

Event Queuing

When an ObjEvent icon executes it looks at a single event and places the appropriate values in the system variables. Because several events can occur before the next ObjEvent icon executes, IconAuthor needs to keep track of these events. Behind the scenes, IconAuthor makes sure that no events are lost by storing them, in the order in which they occur, in the **event queue**.The event queue is essential to those situations where you define an ObjEvent icon to *check* for an event. Remember that in this situation, the icon either finds an event, handles it, and proceeds or it doesn't find an event and proceeds anyway. For example, you can put the ObjEvent icon in a loop where a series of icons executes repeatedly and flow continually returns to the ObjEvent icon to check for an event. A user or an object (such as a Timer) may generate an event; regardless of the source, each time an event occurs it is placed in the queue. Every time the ObjEvent icon executes it takes the oldest event off the queue, stores the appropriate information in the system variables and continues execution.

Command

Specify whether the ObjEvent icon should wait for an event to occur by using either a Wait or No Wait command. Acceptable values are: Wait, No Wait, or a variable.

Drop-down List Box Items:

Wait - Indicates that you want the icon to wait (either indefinitely or for a particular period of time) for an event to occur before execution flows to subsequent icons.

No Wait - Indicates that you want the icon to check to see whether it detects an event. If it finds that an event has occurred, it stores information about that event in the system variables and execution flows downward. If it does not find an event, execution simply flows downward.

Variable Selector

Timeout

If you specified Wait as the Command, use this text box to indicate how long the icon should wait for an event to occur. Acceptable values are: 0, a positive integer, or a variable.

Specifying 0 means that you want the ObjEvent icon to wait indefinitely for an event. Specifying an integer indicates the number of seconds you want the icon to wait for an event.

Drop-down List Box Item:

Variable Selector

ObjGet Icon

The ObjGet icon retrieves the current setting of an object's property and stores that value in a variable. The object can be a live object created and displayed via a SmartObject file or it can be the window object.

Retrieving the Property of an Object

In most situations, the ObjGet icon is used to retrieve the property setting of a specific object. For example, if a user types text into a text block, the input becomes the setting for the Text property of that object. Use an ObjGet icon to retrieve that property setting and store it in a variable. Your structure can evaluate the value stored in the variable and branch accordingly.

Retrieving the Property of a Family of Objects

There is one situation where you use the ObjGet icon to retrieve a property shared by an entire family of objects. This property is called CheckedRadioButton. When the family is made up of a group of radio buttons you use an ObjGet icon to retrieve the CheckedRadioButton of the family to determine which option the user selected.

Suggested Uses:

- Give the user the opportunity to type text in a text block. The text that the user types becomes the current setting of the Text property for that text block. Use an ObjGet icon to retrieve the Text property setting and store it in a variable. You can now display, manipulate, and/or evaluate the text input that is stored in the variable.
- Give the user the opportunity to select one of several items (for example, colors) from a list box. When a user clicks on an item (such as the color blue) the SelectedItemData property of the list box is set to that selection (blue). Use an ObjGet icon to retrieve that property setting (blue) and store it in a variable such as @color. Your structure can evaluate the value stored in @color and branch accordingly.

Content Editor Text Boxes:

Scope
Name
Property
Variable Name

Scope

Specify whether you want to retrieve a property of one object or a family of objects. Acceptable values are: Object, Family, or a variable.

Drop-down List Box Items:

Object - Indicates that you want to get the property of a single object.

Family - Indicates that you want to get the property (CheckedRadioButton) for a family of objects.

Variable Selector

Name

Specify the name of the object or family (or a variable).

Drop-down List Box Items:

Object Name Selector
Variable Selector

Property

Specify the property you want to retrieve. Choose Object Property Selector from the drop-down list box to help you find the property you want. Click on the drop-down list box of the Properties box to display the list of object classes. Choose the appropriate class. The list box below shows the properties available for the chosen class. Choose a property and choose OK to return the property to the Content Editor.

Drop-down List Box Items:

Object Property Selector - Accesses the Object Property Selector which lets you select a property for this text box. In the drop-down list box at the top of the dialog box, choose the appropriate object class. For example, if you are getting a list box property, choose List Box. Once you choose a class the selection of properties in the list box below changes to reflect your selection. Click on the property you want to get and choose OK. The property is automatically returned to the Content Editor text box.

Variable Selector

Variable Name

Specify the variable in which you want to store the current property setting.

Drop-down List Box Items:

Variable Selector

ObjMenu Icon

The ObjMenu icon is a composite that makes it easier for you to use live objects to create a part of an application that presents the user with a menu screen. The menu displays live objects (via a SmartObject file) that represent several activities or options. Once the objects are displayed a loop structure executes and an ObjEvent icon awaits the user's interaction. When the user makes a choice from the menu, a branch is taken, an action occurs, and then the loop begins again. This process continues until the user exits the loop (until an Exit icon is reached).

The ObjMenu composite is a framework. In most cases you would need to edit the content of the icons, and minimally, you would need to add other icons to the structure to make it perform usefully. As an example, by default there are four If icons, labeled **1**, **2**, **3**, and **Exit** in the composite. Your application may need a smaller or larger number of branches. You can cut, copy, paste, and edit the content of If icons to create the required number of branches.

Also, you must add the icons that cause a particular action to occur when a branch is taken. Under each If icon, you add a series of icons to fulfill the user's selection. For example, if the first choice is labeled "Tutorial" and is selected from a menu, the icons in the first branch must present a tutorial.

The following list describes the function of each icon in the ObjMenu composite:

1. **ObjMenu icon**: Marks the beginning of the ObjMenu composite. The Content Editor contains only a Composite Name text box to let you customize the name of the composite.
2. Display icon**:** Displays the SmartObject page that is the menu. By default, the composite assumes the following:

+ The display consists of four live objects (one for each If icon in the Branches composite).
+ Each object on the page has an ObjectName. Three objects have the names "1," " 2," and "3," respectively and the fourth object is called "Exit."
+ Each object also has a Notify- property set to True so that when the user interacts with the object, IconAuthor will be notified. For example, if one of the objects is a Push Button and its NotifyOnClickLeft property is True, IconAuthor will be notified when the user clicks left on the object.

3. **LoopStart icon**: Marks the beginning of the loop within the ObjMenu composite. Does not have a Content Editor.
4. ObjEvent icon: Causes the application to wait for the user to interact with one of the live objects. As soon as a user interacts, the name of the selected object is stored in @_Object_Name and the name of the event that occurs is stored in @_Object_Event. For example, if the user clicks left on a button called Exit, @_Object_Name = Exit and @_Object_Event = ClickLeft.
5. Branches Icon: Marks the beginning of the Branches composite within the ObjMenu composite. The Branches composite causes the application to take a particular branch or path of execution depending upon the object the user activated.
6. **1, 2 3, and Exit icons**: The four If icons built into the ObjMenu composite. Each of these icons marks the beginning of a different branch. After the user interacts with a live object, the **1 icon** compares the value of @_Object_Name to the number 1. If @_Object_Name = 1, the condition is true, and execution flows downward to any icons built into the first path. If @_Object_Name $\neq$ 1, execution flows to the right to the **2 icon**. Each of the numbered If icons makes a similar comparison until one of the conditions is true.

   If the **1 icon**, **2 icon**, or **3 icon** test true, the corresponding branch is executed. After the branch completes, the loop begins again. The live objects that were displayed via the Display icon (prior to the loop) are still displayed. They continue to be displayed until they are deleted or hidden. If the **Exit icon** tests true, the corresponding branch (the Exit icon) is executed.
7. Exit Icon: This is the only icon in the fourth branch. It allows the user to exit from the loop composite. (Without an explicit exit, this ObjMenu composite loop, like the composite Loop icon, would execute

infinitely.) "Loop" is specified in the Exit From text box. When this icon is executed, execution flows out of the loop to the ObjDelete icon.

8. **LoopEnd icon**: Marks the end of the loop. This icon is executed if one of the first three branches (icon **1**, icon **2**, or icon **3**) is executed. After the LoopEnd icon, execution returns to the beginning of the loop within the Menu composite.
9. ObjDelete Icon**:** This icon is executed only when the user interacts with the object called Exit. It deletes all the live objects from memory.

ObjQueue Icon

Use the ObjQueue icon to control the event queue. An event is an *action, involving live objects, that is recognized* by your application. An example of an event is a mouse click on a button or a Timer count down that reaches 0. IconAuthor keeps track of events by storing them, in the order in which they occur, in the event queue. In some situations, it is important that every event is stored in the queue. For example, an ObjEvent icon in a loop can be defined to look for events generated by a user and/or by a Timer. In this situation, if the user clicks and the Timer goes off, you want both events to go in the queue so they can be handled, each in their turn, by the ObjEvent icon.

There are other cases where you would use the ObjQueue icon to remove the events currently in the event queue. In effect, you are clearing or flushing the queue so that it starts over empty. As an example, when your application begins, it may be important to store all events in the event queue. At some point, a sub-application is called by the main application. When the sub-application runs, it displays another SmartObject page and then an ObjEvent icon waits for an event. You don't want any "left over" events that are still in the queue to be processed in conjunction with the new objects. Therefore, the first icon in the sub-application is an ObjQueue icon with an Empty command.

Content Editor Text Boxes:

Scope

Scope

Specify the action you want to affect the event queue. Acceptable values are Empty or a variable.

Drop-down List Box Items:

Empty - Indicates that you want to clear all events from the event queue.

Variable Selector

ObjSet Icon

Use the ObjSet icon to change the property of an object, a class of objects, a family of objects, or All objects. An object is a live object created and displayed via a SmartObject file or the window object.

Suggested Uses:

- Your application can provide the user with a button labeled "Start" for starting a video. As soon as the video begins playing, use an ObjSet icon to change the Text property of that same button to "Stop." Now let the user click on the Stop button to stop the video at any time.
- Your application can contain a graphic object that initially shows a picture of one view of an automobile. When the user clicks on a button to indicate that they want to change to a different view, an ObjSet icon can change the Filename value of the graphic object so that a different picture is displayed.
- If your application frequently uses an OK button and Cancel button, always together, make them one family called OkCancel. At runtime when your application is not using these buttons, use an ObjSet icon to make the Visible property of every object in the OkCancel family *not* visible.

Content Editor Text Boxes:

Scope
Name
Property
Value

Scope

Specify whether you want to set the property of one object, a family or class of objects, or All objects. Acceptable values are: Object, Family, Class, All, or a variable.

Drop-down List Box Items:

Object - Indicates that you want to set a property for a single object.

Family - Indicates that you want to set a property for a family of objects, for example, all objects that have their FamilyName property set to ColorButtons.

Class - Indicates that you want to set a property for an entire class of objects, for example, all buttons or all OLE objects.

Page - Indicates that you want to set a property for an entire page of objects.

All - Indicates that you want to set a property for all live objects.

Variable Selector

Name

Specify the name of the object, class, or family (or a variable) for which you want to set a property.

Drop-down List Box Items:

Object Name Selector

Object Class Selector - Accesses the Object Class Selector which lets you select the name of a class for this text box. Click on the class for which you want to set a property and choose OK. The class is automatically returned to the Name text box.

Variable Selector

Property

Specify the property you want to set (or a variable). Choose Object Property Selector from the drop-down list box to help you find the property you want to use.

Drop-down List Box Items:

Object Property Selector - Accesses the Object Property Selector which lets you select a property for this text box. In the drop-down list box at the top of the dialog box, choose the appropriate object class. For example, if you are setting a list box property, choose List Box. Once you choose a class the selection of properties in the list box below changes to reflect your selection. Click on the property you want to set and choose OK. The property is automatically returned to the Content Editor text box.

Variable Selector

Value

Specify the value to which you want to set the specified property. The drop-down list box provides frequently used values, such as True and False. It also lets you access a number of editors and dialog boxes to select a value such as a color, font, or screen location.

Drop-down List Box Items:

True - Use this value to set a property such as Enable or Visible to true.

False - Sets a property such as Enable or Visible to false.

Area Editor - Accesses the Area Editor which lets you visually select an area on the screen. When you close the Area Editor, values that define the selected area are returned to the Content Editor.

Location Editor - Accesses the Location Editor which lets you visually select a location (a point) on the screen. When you close the Location Editor, the coordinates of the selected point are returned to the Content Editor.

Directory - Accesses a Directory file selection dialog box. Use this dialog box to select a filename. When you choose OK the dialog box is closed and the filename is automatically returned to the Content Editor.

Color Editor - Accesses the Colors dialog box where you can select a color or create a custom one. This dialog box is used in several capacities by IconAuthor and its Editors.

Solid Color Editor - Accesses the Solid Color dialog box where you can select a solid color. When you choose OK, the color is automatically returned to the Content Editor.

Font Editor - Accesses the Font dialog box which lets you choose a font, size, and color for text. This dialog box is used in several capacities by IconAuthor and its Editors.

Variable Selector

Parse Icon

The Parse icon parses (breaks down) a string into its individual words or characters, and places the words or characters into an array.

Suggested Uses:

- Parse a user's response to a question to determine if it includes the word that is the correct answer
- Parse different pieces of information previously stored in an indexed variable where each piece of information is separated by a specific delimiter.
- Parse a string, change some of the characters or words in each parsed element, and use a Variable icon to concatenate the changed characters or words back into a string.

Content Editor Text Boxes:

String To Parse
Word Count Variable
Word Delimiter
Array Name

String To Parse

Specify the variable that contains the string you want to parse. (You can also specify a fixed value for this field, such as "Hello George", although most often you will want to parse a value stored in a variable.)

Drop-down List Box Items:

Variable Selector

Word Count Variable

Specify the variable in which you want to store the number of elements the parsed string is broken down into. For example, if you parse "Hello George" into separate *words*, and specify @WORD_COUNT in the Word Count variable text box, @WORD_COUNT will contain the value 2 after the parse icon is executed. As another example, if you parse "Hello George" into separate *characters*, and specify @CHAR_COUNT in the Word Count variable text box, @CHAR_COUNT will contain the value 12 (for the 11 letters and 1 space in "Hello George").

Drop-down List Box Items:

Variable Selector

Word Delimiter

Specify how you want to break down the string being parsed. If you want to parse the string into words, specify the characters IconAuthor should interpret as delimiters. For example, if the string being parsed is a sentence, include basic punctuation characters in this text box (**,.?,!;:**).

In some situations, the string being parsed is several pieces of information that are deliberately stored in a variable using a specific delimiter, such as a tilde (~). In this case, you can specify the known delimiter in the Word Delimiter text box.

If you want to parse the string into individual characters (instead of words), specify "character" in the Word Delimiter text box.

Acceptable values are: a punctuation mark (for example **.?,!:;~**), a space, any other displayable character or characters, or a variable.

Note: If you want to use a space as a delimiter, simply press the space bar to enter a blank space in this field.

Drop-down List Box Items:

character - Causes the string to be parsed into individual characters.

Variable Selector

Array Name

Specify the name of the array (indexed variable) that will store the resultant elements when the string is parsed. (Specify array variables without brackets.) For example, if the string "Hello George" is parsed into words, and @ELEMENT is specified in the Array Name text box, @ELEMENT[1] = Hello and @ELEMENT[2] = George, after the Parse icon is executed. If "Hello George" is parsed into individual characters, @ELEMENT[1] = H, @ELEMENT[2] = e, and so on.

Drop-down List Box Items:

Variable Selector

Pause Icon

The Pause icon causes execution to pause for a specified period of time.

Number of Seconds text box:

Specify the number of seconds for which you want to pause execution.

Acceptable values are: a real number greater than 0, such as 3 or .5; or a variable.

Drop-down List Box Items:

Variable Selector

Print Icon

The Print icon prints a file or a current screen display. It uses the printer currently selected through Microsoft Windows.

Suggested Uses:

- Allow the user to create an image on the screen and print a copy of the image.
- Create a course that provides instruction to the user, and give the user the option of producing a printed copy of portions of the course.
- Let the user choose an image to be printed, for example a coupon or flyer on a particular product.
- Provide the user with a printed receipt.

Content Editor Text Boxes:

File Type
Filename
Location
Page

File Type

Specify the information that you want to print. This can be a file or the current screen.

Acceptable values are: **bitmap**, **screen**, **smartobject**, **formatted text**, or a variable.

Drop-down List Box Items:

bitmap - Indicates that you want to print a bitmap graphic, such as a file created with Paintbrush. Select this item if you are printing a graphic with one of the following formats:

.ATT	.FIF	.KFX	.PSD	.XPM
.BMP	.GIF	.LV	.RAS	.XWD
.CAL	.GX2	.MAC	.RLE	
.CLP	.ICA	.MSP	.TGA	
.CUT	.ICO	.PCD	.TIF	
.DCX	.IFF	.PCT	.WMF	
.DIB	.IMG	.PCX	.WPG	
.EPS	.JPG	.PIC	.XBM	

screen - Indicates that you want to print the current screen.

formatted text - Indicates that you want to print a formatted text file with an .FTT, .RTF, or .TXT extension.

Note: IconAuthor (via the SmartObject Editor) supports only rudimentary .RTF file formatting, such as font, font size, and font color information. If you attempt to display an .RTF file with more complex formatting information, such as tables, columns, or embedded graphics, the SmartObject Editor will not display this information. For example, any text that is part of a table will not display.

SmartObject - Indicates that you want to print a SmartObject file, created with the SmartObject Editor.

Variable Selector

Filename

If you are printing a file, specify the name of the file you want to print. If you are printing the current screen, leave this text box blank.

Drop-down List Box Items:

Directory
SmartObject Editor
Variable Selector

Location

Specify the location of the file you want to print. The Location text box lets you choose coordinates for the *upper left corner* of the file being printed. If the file is smaller than the full screen, you can choose coordinates that place the file anywhere on the printed page. The default, 0,0 places the file in the upper left corner of the page. You can use other coordinates that center the file, or have it appear in a particular corner of the page.

If you choose negative coordinates for upper left corner of the file, or extremely large coordinates, part or all of the file may not appear on the page. Negative coordinates may place the file very high (off the page) or to the left (off the page). Large coordinates may place the file very low (off the page) or to the right (off the page).

Note: IconAuthor uses dimension information contained in a file to place the file on the page. Windows Metafiles do not contain dimension information, therefore, .WMF files that you print may not appear as expected. Run an application that contains .WMF files to check to see whether each file is printed satisfactorily.

Acceptable values are: a pair of screen coordinates or a variable.

Drop-down List Box Items:

Location Editor
Variable Selector

Page

If you are printing a page in a SmartObject file, use this text box to specify page name or number.

Program Icon

The Program icon runs an external program from your IconAuthor application. When the new program finishes execution, the icon below the Program icon is executed.

When the other program is running, one of two things can happen:

+ The program being executed is in complete control of the system; IconAuthor has no control until the user deliberately exits the program and control is passed back to IconAuthor.
+ The program and IconAuthor have control of the system simultaneously and IconAuthor can continue to run regardless of the status of the program.

If the Program icon runs a program such as Calculator, users can manipulate the calculator in the normal manner. When they are finished using the calculator, they must choose Exit from the File menu of the Calculator window or double-click on the Control Menu box to exit from the application.

When the new program is run, you can specify that IconAuthor remain in view (as a full screen background for the new program). If the IconAuthor application is being run in a window, you can specify that it remain on the screen in the current window. You can remove IconAuthor from the screen completely, or you can cause only a small area of the current IconAuthor application screen to be visible.

Warning: If IconAuthor remains as a full screen background when the new program runs, the user *must not* minimize the new program. If the new program is minimized, the user must press ALT + ESC to open the Windows Task List, select the program that they minimized, and choose OK. The program is redisplayed and once again available for use. It must be exited by the normal exit procedure in order to return control to the IconAuthor application.

Suggested Uses:

- Create an IconAuthor application that teaches a user to use another application, such as dBASE or Excel. Use the Program icon to run the application being taught, to let the user gain practical experience.
- Create an IconAuthor application that lets users access other applications to accomplish certain tasks (for example, use a Calculator, or a database).

Content Editor Text Boxes:

Program Name
Window Size
Snapshot?
Wait Until done?

Program Name

Specify the path and the name of the program you want to run from IconAuthor. For example, if you want to run the Windows Notepad text editor and it is located in the WINDOWS directory of your C: drive, specify C:\WINDOWS\NOTEPAD.EXE.

Optionally, you can also specify the name of a particular file to be opened when the new program is run. For example, if you specify C:\WINDOWS\NOTEPAD.EXE MYFILE.TXT, the Program icon will run Notepad and open the file MYFILE.TXT.

Acceptable values are: a filename specification (including directory path, executable filename, and optionally, a data filename), or a variable.

Note: If a path is not specified, the IconAuthor install directory is used.

Drop-down List Box Items:

Directory
Variable Selector

Window Size

Specify the size of the window in which the IconAuthor application appears when the new program runs.

Note: This setting does not control the size and position of the new program you are running. These factors are controlled by Windows.

Acceptable values: **current size**, **full screen**, **remove**, 4 numbers (separated by commas) that define a rectangular area on the screen (the first 2 numbers are the x, y coordinates of the upper left corner of the area, the second 2 numbers are the width and height of the area), or a variable.

Drop-down List Box Items:

current size - The IconAuthor application remains in the its current window size and position, even if its current size is full screen.

full screen - The IconAuthor application remains a full screen background for the new program

remove - The IconAuthor application is removed from the screen. Note that unless the new program comes up full screen, the IconAuthor window or other portions of the Windows environment are visible in the background.

Area Editor

Variable Selector

Snapshot?

Specify whether you want the IconAuthor application context to be restored after the user exits the program run by the Program icon. The application context is the information displayed on the screen, the current video frame number, the audio channels, and the overlay mode.

Acceptable values are: **no**, **yes**, or a variable.

Drop-down List Box Items:

no - The context *is not* saved. When the new program is exited, the IconAuthor application continues with the screen, video, and audio settings as they are.

yes - The context *is* saved. When the new program is exited, the IconAuthor application resumes with the screen, video, and audio settings as they were before the Program icon was executed.

Variable Selector

Wait Until Done?

Specify whether you want the IconAuthor application to continue executing while the new program is executing.

Acceptable values are: **no**, **yes**, or a variable.

Drop-down List Box Items:

no - The IconAuthor application will continue executing while the new program is executing.

yes - The IconAuthor application stops executing, resuming only when the new program finishes.

Variable Selector

RS-232 Icon

The RS-232 icon permits the system on which an IconAuthor application is running to communicate with a peripheral device such as a modem, videodisc player, mainframe, mouse, or printer. By issuing commands through a series of RS-232 icons you can open, configure, send data and receive data through an RS-232 communications port.

Each RS-232 icon issues a command string through the Control text box. You can combine several commands within one string, however, a command that sends data must be issued in a separate icon from one that receives data. If the command sends data through the port, the Transmit Data text box can contain a string to be transmitted or a variable whose value will be transmitted. If the command specifies data received through the port, the received information is stored in a variable you enter in the Receive Variable text box.

Before you can send or receive information, you have to open a port (with a /p command). If the port is not open, all transmitted information is discarded.

Also, before your system can communicate with a peripheral device, you may have to set communication parameters for the port so that it meets the requirements of the peripheral device. The default communication parameters are:

Speed	1200 bits per second
Parity	None
Data bits	8
Stop bits	1

To determine whether you need to reset these parameters, refer to the documentation that accompanies the peripheral device you are using.

Content Editor Text Boxes:

Control
Transmit Data
Receive Variable

Control

Specify the command string that controls how data is sent or received. Use commands to open a port (/p), set communications parameters for a port (/c), send data through a port (/S or /s), and receive data through a port (/R or /r). Another command, the timeout command (/t), can set the period of time in which data can be received.

Acceptable values are: a command string (send and receive commands must be issued in separate RS-232 icons), or a variable.

The following table describes the available commands:

command and syntax	purpose
/Sxx...	Sends the characters following the /S out the port. The characters are sent in a hexadecimal format. This means for example, that if you want to send the character "A", specify 41.
/s	Sends the characters in the "Transmit Data" text box out the port. The character format is ASCII. If you specify "Hello George", the ASCII equivalent of each character in the phrase will be sent.
/R##	Receives ## (decimal) characters from the port and stores them in the variable specified in the "Receive Variable" text box. Characters are accepted until the maximum receive time (see the /t command) is reached.
/rxx	Receives characters from the port and stores them in the variable specified in the "Receive Variable" text box. Characters are accepted until the hexadecimal character cc is received.
/t##	Controls the default receive time out period. If unspecified, its value is 1 second. The maximum allowable value is 32767. Each /r and /R command will receive information for no longer than the timeout period.
/p#	Specifies the port. /p1 sets the port to COM1 and /p2 sets the port to COM2. In all cases, if the specified port is not already opened and if the port is available, the port is opened. (If the port is not available because it is not present or already in use, all subsequent commands fail.)
/c####,$,%,&	Sets the communications parameters. When the port is first opened, it is set to 1200 bits per second, no parity, 8 data bits, and 1 stop bit. #### (the baud rate) can be **150**, **300**, **600**, **1200**, **2400**, **4800**, or **9600**. $ (the parity) can be **n** (for none), **e** (for even), or **o** (for odd) % (the number of data bits) can be **5**,**6**,**7**, or **8**. & (the number of stop bits) can be **1** or **2**.

Drop-down List Box Items:

Variable Selector

Transmit Data

Specify the data you want to transmit through the port if you are using a /s "send" command in the Control text box. (If you use a /S "send" command in the Control text box, the data being sent is expressed in hexadecimal format as part of the command string.)

Acceptable values are: a character string or a variable.

Drop-down List Box Items:

Variable Selector

Receive Variable

Specify the variable to which you want to store data that is received through a port when you use a /r or /R "receive" command in the Control text box.

Drop-down List Box Items:

Variable Selector

Random Icon

The Random icon generates a random number, from within a specified range, and stores it to a variable.

Suggested Uses:

- When a file used by a training course contains several questions, and you only want to use one, use a Random icon to randomly choose one question
- Create a guessing game that lets a user guess a randomly selected number
- Use a Random icon as part of an "attractor loop". An attractor loop is designed to attract a user's attention. For example, an application can let a user choose to view any of four demonstrations. If however the system is unattended for more than 15 seconds, a Random icon automatically picks one of the four demonstrations, and that part of the application is executed.

Content Editor Text Boxes:

Variable Name
Start Range
End Range

Variable Name

Specify the variable to which you want to assign the randomly generated number.

Drop-down List Box Item:

Variable Selector

Start Range

Specify the low end of the range from which you want a number to be randomly generated.

Acceptable values are: an integer (positive or negative), or a variable.

Drop-down List Box Item:

Variable Selector

End Range

Specify the high end of the range from which you want a number to be randomly generated.

Acceptable values are: an integer (positive or negative), or a variable.

Drop-down List Box Items:

Variable Selector

SaveVar Icon

The SaveVar icon lets your application save one or more application variables and the values they contain to a variable file. A variable file is an ASCII file. After it is created, the variable file can be used by another IconAuthor application, or it can be viewed through a text editor, such as Notepad. (At some later point, to load the variables you have saved using a SaveVar icon, use the LoadVar icon.)

The information you save using the SaveVar icon can be quickly retrieved at a later point in time. The SaveVar icon is therefore particularly useful when you want to save a small amount of information without creating a database.

When you use a SaveVar icon, you can:

- Save all the application variables currently in memory to the variable file.
- Add an application variable to the end of the variable file.
- Save one application variable to a variable file (erase any other information if the file existed previously).
- Update the value of an application variable in an existing variable file.

The format for variable files is the variable name on one line, and the value assigned to the variable on the next line.

Content Editor Text Boxes:

Filename
Control
Variable Name

Filename

Specify the name of the file to which you want to save one or more variables. The file may or may not already exist. If it does not exist, it is created.

Acceptable values are: a filename with a .VAR extension, or a variable.

Drop-down List Box Items:

Directory
Variable Selector

Control

Specify whether you want to save all or one of the current application variables, or if you want to update the value of an existing variable.

Acceptable values are: **all**, **append**, **single**, **update**, or a variable.

Drop-down List Box Items:

all - Saves all the current application variables to the file specified in the Filename text box.

append - Appends one variable to the end of the file specified in the Filename text box. Note that the variable is not inserted in alphabetical order, but is added to the *end* of the file. Also, if you append the same variable twice, the file contains two occurrences of that variable. If the variable file that you are appending to was created with the Notepad text editor, be sure to check that a carriage return follows the last variable entry. An invalid variable entry is created if there is no carriage return.

single - Saves one variable to the file specified in the Filename text box. The file has just one variable after the operation. The previous contents of the file are erased.

update - Updates the value of a variable specified in the Filename text box. If the variable is not in the file prior to this operation, it is added to the file. After the operation, all the variables in the file are in alphabetical order. (If the file contains two or more variables with the same name, all are deleted except for the first one in the variable list.)

Variable Selector

Variable Name

Specify the name of the variable you want to save when you use a single, append, or update command in the Control text box.

Drop-down List Box Items:

Variable Selector

Shuffle Icon

The Shuffle icon randomly rearranges the contents of an indexed variable.

Suggested Uses:

- Shuffle 20 questions when you only want to ask the user 3 of them. Each element of an indexed variable called @QUESTION contains a question. Shuffle the contents of the array, and ask the user @QUESTION[1], @QUESTION[2], and @QUESTION[3].
- As part of a game or test, use the Shuffle icon to jumble the order of letters in a word, or words in a sentence.
- In an application that uses a deck of cards, use the Shuffle icon to shuffle the deck.

Shuffle Array text box:

Specify the name of the indexed variable whose contents you want to shuffle.

Drop-down List Box Items:

Variable Selector

Snapshot Icon

Include a Snapshot icon (set to "on") as the first icon in a Help application. This icon takes a "snapshot" of the screen context (information such as the current video frame, graphic display, etc.) when the Help application is called. When a user is finished using a Help application and chooses to return to the main application, the Snapshot icon restores the context of the application.

Take Snapshot text box:

Specify whether you want the snapshot feature on or off.

When using the Snapshot icon, set it to "on" before you do a full-page display, so that the screen will be refreshed properly. When testing an application, it is helpful to set the Take Snapshot text box to "off" because it takes a little longer to restore the screen context if it is on.

Acceptable values are: **on**, **off**, or a variable.

Drop-down List Box Items:

on - The Snapshot icon takes a snapshot.

off - The Snapshot icon does not take a snapshot.

Variable Selector

Related Topic:

Help Icon

Startup Icon

The Startup icon is a special authoring tool that enhances your ability to run an application from a selected icon. It does not have a Content Editor. IconAuthor's Run menu contains the "Application from Selected" command. When you select an icon in the structure and choose this command the application begins executing from the selected icon. (The right Run button on the function ribbon provides this same functionality.)

If you include a Startup icon as the first icon in your application you can use it to identify key icons that you *always* want to execute, even if you are running the application from a selected icon further down in the structure. Regardless of the position of the selected icon, the icons marked by the Startup icon are always executed *first*.

Although the Startup icon is a composite (green), when you build it into your structure notice that it does not yet have icons to the right of it. It is your job to build icons into the empty Startup composite. Build the first icon to the right of the Startup icon. Build subsequent icons below that first icon.

Note: Although the Startup icon is a tool for enhancing the authoring process it does not affect the performance of an application when it is run with the Presentation System. That means that it is not necessary to remove or alter the Startup composite before distributing your application to end-users.

The following icons are useful items to include within a Startup composite:

Window icon
Help icon
DllLink icon

Note: You can put any icon you want in the Startup composite; even another composite.

SubApp Icon

The SubApp icon lets the main application use another IconAuthor application. Depending on the command you specify in the SubApp icon Action text box, another IconAuthor application is loaded into memory, executed, or removed from memory.

The SubApp icon gives global read and write capability to both the main application and the called application. The variables defined by either application can be used by both. When the called application finishes executing, the application from which it was called resumes executing with the icon below the SubApp icon.

Note: If your application executes multiple sub-applications, subroutines, and loops that are not returned from, you must use the SubApp icon's restart command in the Action text box to prevent errors in execution of your application.

Suggested Uses:

- Create modular applications that call other applications where necessary
- Create an application that suits a particular need, and that application can be called by several different main applications
- Create modular applications using a team development style. Different applications used by the main application can be created by different developers.

Content Editor Text Boxes:

Filename
Action

Related Topic:

Exiting from a Called Application

Filename

Specify the name of the application you want to load, execute, or remove from memory. The called application can have a .IW extension like any other IconAuthor application.

Acceptable values are: a filename or a variable.

Drop-down List Box Items:

Directory
Variable Selector

Action

Specify the action you want to perform on the specified filename.

Acceptable values are: **execute**, **load**, **remove**, **restart**, or a variable.

Drop-down List Box Items:

execute - Loads the specified file into memory (if it is not already loaded) and executes it immediately. At the end of execution, the file remains in memory until it is removed with another SubApp icon. When the called application finishes executing, the application from which it was called resumes executing with the icon below the SubApp icon.

load - Loads the specified file into memory. This option lets you load a called application at the beginning of the main application. The called application is then available for later use and remains in memory until it is removed with another SubApp icon.

remove - Removes the specified file from memory.

restart - Resets system memory to prevent a stack overflow.

Variable Selector

Exiting from a Called Application

There are two ways to exit from a called application. The called application can simply complete execution when it runs out of icons, or it can contain one or more Exit icons

If you specify "subapp" in the Exit From text box of the Exit icon, execution returns to the main application from which the other application was called. The main application resumes execution with the icon below the SubApp icon.

If you specify "application" in the Exit From text box of the Exit icon, this causes an exit from the called application *and* the main application.

Related Topics:

SubApp Icon
Exit Icon

SubAssign Icon

The SubAssign icon is the first icon in a **subroutine**.

In the Accept List text box of the SubAssign icon, you specify the variable or variables in which you want to receive the information being passed from the Subroutine icon in the main application. The number of parameters being passed from the Subroutine icon must match the number of parameters being received in the SubAssign icon.

Content Editor Text Box:

Accept List

Accept List

Specify the variable or variables that will accept the values passed from the Subroutine icon in the main application. These are the values that are manipulated within the subroutine.

The parameters passed from the main application are accepted in the order in which they are passed. For example, you can specify @STUDENT,17 in the Parameter List text box of the Subroutine icon (where @STUDENT contains "Bob Smith"), and you can specify @NAME,@NUM in the Accept List text box of the SubAssign icon. The result is that within the subroutine, @NAME contains "Bob Smith" and @NUM contains the literal "17".

The number of parameters being accepted in the SubAssign icon must match the number of parameters being passed from the Subroutine icon. For example, if you enter the three parameters "@NAME,@NUM,6" in the Parameter List text box of the Subroutine icon, the Accept List text box of the SubAssign icon must contain a three item entry such as @EMPLOYEE,@NUM,@MONTHS".

Acceptable values are: one or more variables.

Drop-down List Box Item:

Variable Selector

Related Topic:

Subroutine Icon

Subroutine Icon

The Subroutine icon allows an IconAuthor application to use a **subroutine**.

Content Editor Text Boxes:

Filename
Parameter List
Receive List
Action

Related Topics:

Loading, Executing and Removing Subroutines
Subroutine Execution

Loading, Executing, and Removing Subroutines

Use Subroutine icons to accomplish the following:

- Load the subroutine into memory.
- Execute the subroutine.
- Remove the subroutine from memory.

A subroutine can be loaded into memory and stay resident until it is used (using a "load" command), or it can be loaded into memory and used immediately (using an "execute" command). If a subroutine is already loaded into memory, it can be executed using an "execute" command.

When a subroutine is a small structure, it is typically loaded and executed immediately the first time it is required by the application. However, when a subroutine is large, it is recommended that it be loaded at the beginning of the application. Because a larger subroutine may cause a delay of up to several seconds when it loads, it is less intrusive when the delay occurs at the beginning of the application, rather than in the middle.

When the application no longer needs a subroutine, you can use another Subroutine icon to remove a subroutine from memory. When you remove a subroutine from memory, you also remove the variables referenced by that subroutine.

Variables in Subroutines

The variables of the subroutine and the main application are separate. The subroutine can use variables of the same name as those in the main application, and yet the values of the variables in the main application are unaffected.

Related Topic:

Subroutine Icon

Subroutine Execution

When a Subroutine icon specifically executes a subroutine, the icon acts as the sender of information. If the subroutine passes any values back to the main application, the same Subroutine icon also acts as a receiver of information.

The subroutine itself contains a SubAssign icon that accepts information from the Subroutine icon in the main application. Other icons in the subroutine manipulate the data accepted by the SubAssign icon. Ultimately, the subroutine also contains at least one Exit icon that when executed, returns the flow of execution to the main application. If the subroutine is passing information back to the main application, the Exit icon is the icon that passes that information.

In the Parameter List text box of the Subroutine icon, you specify the information you want to send to the subroutine. This information can be a single value application variable, an indexed variable, or a literal (a word, letter or number with no spaces).

The first icon in the subroutine being executed is a SubAssign icon. In the Accept List text box of the SubAssign icon, the variable or variables that will receive the information being passed from the main application are specified. The number of parameters being passed from the Subroutine icon must match the number of parameters being received in the SubAssign icon.

When the subroutine has performed its task, it is exited via an Exit icon that contains "subroutine" in the Exit From text box. Also, the Return List text box of the Exit icon, contains the variables used to pass values back to the main application. (Note that some subroutines *do not* pass values back to the main application. A value may already be calculated and displayed within the subroutine itself, and its value may not be important to the main application.)

If the Exit icon does pass one or more variables back to the main application, those values are received by the variables in the Receive List text box of the Subroutine icon. The number of parameters being passed from the Exit icon must match the number of parameters being received by the Subroutine icon (in the main application).

Subroutines may be nested up to a depth of ten. Up to 64 subroutines can be in memory at any given time.

Hint: Once you create a subroutine, you can test it by supplying it with simple values which will produce an obvious result. After you know the subroutine works properly you can use it repeatedly.

Related Topic:

Subroutine Icon

Filename

Specify the name of the subroutine you want to load, load and execute, execute, or remove.

Acceptable values are: a filename or a variable.

Drop-down List Box Items:

Directory
Variable Selector

Parameter List

Specify the parameters you want to pass to the subroutine. Each piece of information you specify must be separated from the next by a comma. The parameters can be any combination of single value variables, a single indexed variable or an entire array, or literal values (a word, letter, or number, with no spaces).

The parameters passed through this text box are accepted and placed in the variables in the Accept List text box of the SubAssign icon (in the subroutine). The parameters are accepted in the order in which they are passed. For example, you can specify @STUDENT,17 in the Parameter List text box (where @STUDENT contains "Bob Smith"), and you can specify @NAME,@NUM in the Accept List text box of the SubAssign icon. The result is that within the subroutine, @NAME contains "Bob Smith" and @NUM contains the literal "17".

The number of parameters being passed from the Subroutine icon must match the number of parameters being accepted in the SubAssign icon. For example, if you enter the three parameters "@NAME,@NUM,6" in the Parameter List text box of the Subroutine icon, the Accept List text box of the SubAssign icon must contain a three item entry such as "@EMPLOYEE,@NUM,@MONTHS".

In the preceding example the value passed in @NAME is received by a variable with a different name, @EMPLOYEE, and the value passed in @NUM is received by a variable of the same name, @NUM. Remember, although you can use the same variable names in the subroutine that you use in the main application, these variables are separate from the variables of the same name in the main application. In order to affect a variable in the main application, you must pass the value back to the main application (through the Exit icon Return List text box).

To pass an entire indexed variable to a subroutine, specify only the name of the variable. For example, if an indexed variable called @NAME in the main application, is made up of @NAME[1], @NAME[2], and @NAME[3], and you want to pass all of these elements to a subroutine, specify @NAME in the Parameter List text box. In the Accept List text box of the SubAssign icon, specify the name of the indexed variable you want to accept these elements. For example, if the indexed variable @NAME is passed to a SubAssign icon that contains @STUDENT in its Accept List text box, the SubAssign icon automatically creates @STUDENT[1], @STUDENT[2], and @STUDENT[3].

To pass individual elements of an indexed variable to a subroutine, specify the name and index of the variable in the Parameter List text box. For example, to pass @COLOR[1] and @NUM[3] to a subroutine, specify these elements (including their name and index). In the Accept List text box of the SubAssign icon, specify the variable you want to accept these values. Because you are passing individual values that are part of an indexed variable, the values can be accepted in a variable that has a name and an index, or in a single value variable name. For example, @COLOR[1] can be accepted in @SHADE[1], @SHADE[6], or @SHADE.

Acceptable values are: a literal (a word, letter, or number with no spaces), or a variable.

Drop-down List Box Item:

Variable Selector

Receive List

If one or more values are going to be returned to the main application from the subroutine, specify the name of the variables in which the values will be received.

If a subroutine passes values back to the main application, they are specified in the Return List text box of the Exit icon (within the subroutine). The number of parameters being passed from the Exit icon must match the number of parameters being received by the Subroutine icon (in the main application). If this text box contains more than one variable, separate each variable from the next with a comma.

Acceptable values are: one or more variables.

Drop-down List Box Item:

Variable Selector

Action

Specify whether you want the Subroutine icon to load a subroutine into memory, execute a subroutine, or remove a subroutine from memory.

Acceptable values are: **execute**, **load**, **remove**, or a variable.

Drop-down List Box Items:

execute - Executes the subroutine specified in the Filename text box. If the specified subroutine was not previously loaded into memory, it is loaded and then executed immediately. After it completes execution, the icon below the Subroutine icon is executed. The subroutine remains in memory until it is removed with another Subroutine icon.

load - Loads the subroutine specified in the Filename text box. This option lets you load the subroutine at the beginning of the application and include another Subroutine icon later in your structure to execute the subroutine. The subroutine remains in memory until it is removed with another Subroutine icon.

remove - Removes the subroutine specified in the Filename text box. This option is helpful if your system has memory limitations.

Variable Selector

Text Icon

The Text icon displays the contents of an ASCII text file on the screen.

To set the font type and/or font size that are used to generate the characters on the screen, precede the Text icon with a Font icon. If you do not use a Font icon, the system default font and size are in effect.

Note: When you use the Font icon, the new settings are in effect until you use a subsequent Font icon.

Content Editor Text Boxes:

Filename
Upper Left Corner
Lower Right Offset

Related Topics:

Font Icon

Filename

Specify the name of the ASCII file you want to display.

Acceptable values are: a filename or a variable.

Drop-down List Box Items:

Directory
Variable Selector

Upper Left Corner

Specify the location of the upper left corner of the file when it appears on the screen.

Acceptable values are: **upper left**, a pair of coordinates (separated by commas) that define the upper left corner of a rectangular area on the screen, or a variable.

Drop-down List Box Items

upper left - Defines the upper left corner of the file as the upper left corner of the screen.

Area Editor

Variable Selector

Lower Right Offset

Specify the width and height of the area where the file appears on the screen.

If the area selected is not large enough to display the entire ASCII file specified in the filename text box, the text is truncated on the screen. It is not scrolled.

Note: A value automatically appears in this text box if you use the Area Editor to specify the upper left corner of the area where the file appears.

Acceptable values are: **lower right**, 2 numbers (separated by commas) that define the width and height of the area where the file appears, or a variable.

Drop-down List Box Items

lower right - Defines the width and height of the area where the file appears, so that the lower right corner of the file appears in the lower right corner of the screen.

Variable Selector

V:Audio Icon

The V:Audio icon controls whether a specific audio channel of the video source is on while a video is playing.

Most videodiscs and videotapes provide sound for two audio channels. This allows for two-channel applications such as those that use stereo, provide music on one channel and voice on another, or those that use different channels for different languages.

In the case of a videodisc player, when both channels are on, each one typically plays through a different audio output jack on the rear of the player. If you turn one channel off, the other channel is played through both audio output jacks.

In the case of a videotape player, it is possible that only one of the channels is available for audio because the other is used for data communications between the player and your system. Note that if you are using a videotape player, the use of channels varies from one machine to another. Refer to the documentation for your videotape player for information on how to use the available channels.

If your source is a digital video file, it may or may not have audio. If the file has audio, turn both channels on to play the audio.

If you do not include a V:Audio icon in your application, all audio channels are set to on. However, you can use one or more V:Audio icons to control which channels are on, and when. Each V:Audio icon can specify an instruction to control one audio channel. For example, you can use one icon to turn one channel off, and then use another icon to turn the other channel off. Subsequent icons can turn the channels on and off again as necessary.

Your application may only require one channel to play at a time, particularly if it uses different languages on different channels. In other situations, you might want to turn the audio off completely. For example, you might not want any audio accompaniment, or you might want to show a video and play audio that is different from the audio that comes with the disc or tape.

Content Editor Text Boxes:

Audio Channel
On or Off

Audio Channel

Specify the audio channel you want to turn on or off.

Acceptable values are: **1**, **2**, or a variable.

Drop-down List Box Item:

Variable Selector

On or Off

Specify whether you want the channel on or off.

Acceptable values are: **on**, **off**, or a variable.

Drop-down List Box Items

Variable Selector

V:Frame#? Icon

The V:Frame#? icon determines the current video source frame number and stores it in a specified variable.

The V:Frame#? icon is particularly useful when you are building an application that lets the user control the video. For example, an application allows a user to stop viewing the video, move to another activity, and resume viewing at a later time. When the user makes a selection to quit the video, the V:Frame#? icon can store the current frame number in a variable, called @RESUME_FRAME. Later in the application, when the user wants to view the video again, a V:Still icon is used to move the video player to the specified frame number. The value in the Frame Number text box of the V:Still icon is @RESUME_FRAME. When the video begins playing again, it plays from the frame number stored in @RESUME_FRAME by the V:Frame#? icon.

Variable Name Text Box:

Specify the variable in which you want to store the current frame number.

Drop-down List Box Item:

Variable Selector

Note: When using a videotape, if a V:Frame#? icon is executed and the videotape player is still in search mode, the string "NOT ARRIVED" is stored to the variable specified in the Variable Name text box.

V:Overlay Icon

The V:Overlay icon has two primary functions. It lets you take advantage of the features of the video overlay card installed in your system. It also lets you specify the name of a file if your video source is a digital video file. (See the source command for the Overlay Command text box for more information on digital video files.)

The overlay card controls how video appears on the screen and in some cases, it controls how audio that accompanies video is heard through your system's speakers. When you send commands to the overlay card, you are not affecting the video source, you are affecting the way video and/or audio appears or sounds. For example, depending on its capabilities, an overlay card might let you freeze an image on the screen. Although the video image appears as a still image on the screen until you issue an unfreeze command, the video player continues to play.

Each V:Overlay icon you use lets you issue a separate command to the overlay card. Different video overlay cards have support for different commands. Refer to the documentation that came with your overlay card for more information on its capabilities.

If you do not include a V:Overlay icon in your application, the overlay mode is set to graph, as if a graph overlay command had been given. Graph means that only graphics are displayed, therefore, in this mode, video is not shown, but the audio that accompanies video is heard.

Depending on the commands supported by your overlay card, you can use a V:Overlay icon to issue either a super or video command so that video is displayed on the screen. Subsequent V:Overlay icons can switch to different overlay modes as necessary.

Content Editor Text Boxes:

Overlay Command
Transparent Color
Parameter 1
Parameter 2

Overlay Command

Specify the command you want to issue to the overlay card. Syntax varies for different commands. For some commands, you only have to enter the command name. For other commands you must provide additional information in the Parameter 1 and Parameter 2 text boxes. The drop-down list box contains some of the available commands. You can also enter a different command in this text box, if it is supported by your overlay card. Refer to the documentation for your particular overlay card to determine which of these commands it supports.

Acceptable values are: an overlay command or a variable.

Drop-down List Box Items:

audio - Turns one or two audio channels on or off. Requires Parameters 1 and 2. Specify on or off in the Parameter 1 text box. Specify 1, 2, or both in the Parameter 2 text box to indicate which audio channel you want to affect. (If your overlay card does not support this command, IconAuthor also lets you control the audio channels through the V:Audio icon.)

capture - Captures the current video frame and stores it to a file. Requires only Parameter 1. Specify the filename in the Parameter 1 text box. Refer to the documentation for your overlay card to learn about file naming conventions for captured files, and to learn about the acceptable entries for file type.

compress - Compresses a portion of the video. Requires Parameter 1 only. Use the Parameter 1 text box to define the area into which you want to compress the video. To do this, enter four numbers separated by commas: the X,Y coordinates of the upper left corner of the area as the first two, and the width and height of the area as the third and fourth respectively.

copy - Copies the current video frame and stores it in a buffer. Requires Parameter 1 only. Use the Parameter 1 text box to define the area of the screen you want to copy. To do this, enter four numbers separated by commas: the X,Y coordinates of the upper left corner of the area, and the width and height of the area. This command is used before a paste command.

crop - Crops a portion of the video. Requires Parameter 1 only. Use the Parameter 1 text box to define the area of the video you want to remain after the cropping is complete. To do this, enter four numbers separated by commas: the X,Y coordinates of the upper left corner of the area, and the width and height of the area.

fade audio - Fades the audio from the current level to a new level. Requires Parameters 1 and 2. Use the Parameter 1 text box to specify the time period (in hundredths of seconds) during which the fade occurs. Use the Parameter 2 text box to enter a value, indicating the level to which the audio should fade (0-100).

fade video - Fades the video from the current level to a new level. Requires Parameter 1 and 2. Use the Parameter 1 text box to specify the time period (in hundredths of seconds) during which the fade occurs. Use the Parameter 2 text box to enter a value, indicating the level to which the video should fade (0-100).

freeze - Freezes the current frame on the screen. Leave Parameter 1 and 2 blank.

graph - Displays only graphics, not video. Use this command to hide the video, and use only the audio. Leave Parameters 1 and 2 blank.

paste - Pastes the video frame currently stored in the buffer, by a copy command, on the screen. Requires Parameter 1 only. Use the Parameter 1 text box to specify the X,Y coordinates of the point on the screen that where you want the upper left corner of the video to appear.

restore - Displays a video frame from a file (previously captured) to the screen. Requires Parameter 1 only. Specify the filename in the Parameter 1 text box. Refer to the documentation for your overlay card to learn about file naming conventions for captured files, and to learn about the acceptable entries for file type.

source - Switches between the available video sources. For example, switch between two players, or

a player and a camera. Requires Parameter 1. Specify the number of the video source to which you want to switch. (If your overlay card supports this command, you would have previously defined the source number that corresponds to each video source.)

Also, use the source command to specify a digital video file. Leave the Transparent Color and Parameter 1 box blank. In Parameter 2, specify the full path and filename of the digital video file to play.

super - Displays video over graphics, everywhere that the transparent color is used on the screen. Use this command to display video and graphics simultaneously. Leave Parameters 1 and 2 blank. Specify the color you want to designate as transparent in the Transparent Color text box.

unfreeze - Unfreezes the video on the screen, by displaying the current frame on the screen. If you use a freeze command to freeze the current video image, that frame appears as a still frame on the screen, and the video player continues to play. When you unfreeze, the video resumes playing at a different point on the disc or tape. Leave Parameter 1 and 2 blank.

video - Displays video only. (Graphics will not appear until a graph or super command is given.) Leave parameter 1 and 2 blank.

Variable Selector

Transparent Color

Specify the color you want to use for the transparent color. This text box is only used when you specify the super command as the Overlay Command. Video appears everywhere that the transparent color appears on the screen.

Acceptable values are: a valid transparent color or a variable.

Drop-down List Box Items:

assorted colors - These are the colors you can designate as the transparent color.

Variable Selector

Parameter 1

When necessary, specify the first parameter for an overlay command. Refer to a specific overlay command to determine whether or not it requires information in this text box, and what kind of information it requires.

Drop-down List Box Items:

Area Editor
Variable Selector

Parameter 2

When necessary, specify the second parameter for an overlay command. See a specific overlay command to determine whether or not it requires information in this text box.

Drop-down List Box Items:

Area Editor
Directory
Variable Selector

V:PlayTo Icon

The V:PlayTo icon plays a video segment from the current frame to the frame number you specify in the Ending Frame text box.

Content Editor Text Boxes:

Ending Frame
Wait Until Done?

Ending Frame

Specify the frame number to which you want the video source to play.

Acceptable values are: a frame number or a variable.

Drop-down List Box Items:

Video Editor
Variable Selector

Wait Until Done?

Indicate whether you want icons to continue executing immediately after the video starts playing, or if IconAuthor should wait until the video is done.

Acceptable values are: **no**, **yes**, or a variable.

Drop-down List Box Items:

no - The icon immediately below the V:PlayTo icon is executed immediately after the video begins playing. Once the video is running, subsequent icons can perform many tasks, such as running an animation routine or displaying a graphic slide show along side of the video. Or, subsequent icons can load subroutines and/or sub-applications that will be executed later in the main application. This is a way of loading them into memory without any noticeable delay to the user. As an example, a V:PlayTo icon can start the video playing. It is followed by a Subroutine icon that loads a subroutine into memory for later use (while the video is playing). Next, a V:Player icon with a wait command causes icons to stop executing until the video segment finishes playing.

yes - The icon below the V:PlayTo icon is not executed until the video segment finishes playing.

Variable Selector

V:Player Icon

The V:Player icon controls the video source.

Each V:Player icon can send a different command to a video source. For example, users can control the video they are viewing. If they select one menu item, labeled FAST FORWARD, a V:Player icon with a fastf command can be executed. If they select another menu item, labeled PLAY, a V:Player icon with a playf command can be executed.

Content Editor Text Boxes:

Control
Receive Data

Control

Specify the command to send to the video source.

Acceptable values are: a command or a variable.

Drop-down List Box Items:

stepf - The first stepf command that is given pauses the source. Each subsequent stepf command steps the source forward one frame.

stepr - The first stepr command that is given pauses the source. Each subsequent stepr command steps the source in reverse one frame.

slowf - Plays forward in slow motion.

slowr - Plays in reverse in slow motion.

playf - Plays forward.

playr - Plays in reverse.

fastf - Plays fast forward.

fastr - Plays fast in reverse.

load - Stops the execution of icons and stops the source to allow for the disc or tape to be checked or changed by the user. Use this command if you want the user to load a tape or disc into a video player. When a load command is executed, a dialog box displays the message, "Please load the video and close the door." The user must load the tape or disc, and respond to the dialog box (by pressing Enter, etc.). The application resumes the execution of icons.

Note: This command may not be supported by your video player.

wait - Stops the execution of icons until the video finishes playing. Use this command if you want to start video, execute one or more icons, and then wait for the video to finish before executing any other icons. For example, in one application a V:PlayTo icon starts the video playing and specifies that it plays until a particular frame. The video shows a person speaking. Next, a Pause icon pauses execution for 2 seconds, although the video is still playing. Then a Display icon displays a graphic over the bottom of the screen that shows the speaker's name. The next icon is a V:Player icon with a wait command. This causes execution of icons to stop and wait for the video to play until the specified ending frame.

Note: This command causes execution of icons to stop until the video finishes playing. Even the designated Escape or Break key cannot interrupt the video.

stop - Stops the video source.

source 1 - If your system is using two video sources, this makes source 1 current. All subsequent V:Player commands are directed to the current video source.

source 2 - If your system is using two video sources, this makes source 2 current. All subsequent V:Player commands are directed to the current video source.

Variable Selector

Receive Data

The Receive Data field is used if your video device is a BCD tape player and is used to receive the value of the tape ID. When a variable is entered in the Receive Data field, a tape ID command is sent to the video tape player and its value, a six digit number is returned to the variable.

Acceptable value: a variable.

Drop-down List Box Item:

Variable Selector

V:Segment Icon

There are two basic ways to create video: use a V:Segment icon, or use a combination of the other IconAuthor video icons. The V:Segment icon plays a video segment until it is finished and does not allow other functions, such as graphic display, to occur while it is playing. Optionally you activate a video interrupt feature to give the user limited control over the material being viewed. The video interrupt feature is a small, on-screen "stop" button the user can select. When the button is selected, it displays a control panel for searching forward or rewinding the segment.

Because the segment played by the V:Segment icon does not allow any other icons to execute until it is finished, it should only be used when a specific portion of your application is dedicated to video. Use the other video icons if you want to display graphics or animation, or load subroutines or other applications, while the video segment is running.

The V:Segment icon Content Editor lets you specify many of the parameters that you would otherwise have to define by using several other icons. Using the V:Segment icon is the fastest and easiest way to include video in your application. However, in some situations, you may want to use the other method of creating video, by building a combination of other IconAuthor video icons into your structure. You might use the other video icons to create your video display if you want to provide the users with custom controls to let them control the video. If you use V:Segment, the on screen buttons and slidebar are fixed and you either use them, or you don't.

Content Editor Text Boxes:

Audio
Overlay Mode
Play From
Play To
Video Interrupt

Related Topic:

Using Video Interrupt

Audio

Specify which audio channels on the video player are active while the video is playing.

Most videodiscs and videotapes provide sound for two audio channels. This allows for two-channel applications such as those that use stereo, provide music on one channel and voice on another, or those that use different channels for different languages. Digital video files may or may not have one channel audio.

In the case of a videodisc player, when both channels are on, each one typically plays through a different audio output jack on the rear of the player. If you turn one channel off, the other channel is played through both audio output jacks.

In the case of a videotape player, it is possible that only one of the channels is available for audio because the other is used for data communications between the player and your system. If you are using a videotape player, the use of channels varies from one machine to another. Refer to the documentation for your videotape player for information on how to use the available channels.

Your application may only require one channel to play at a time, particularly if it uses different languages on different channels. In other situations, you might want to turn the audio off completely.

Acceptable values are: **1**, **2**, **both**, **off**, or a variable.

Drop-down List Box Items:

1 - Turns channel 1 on (turns channel 2 off)

2 - Turns channel 2 on (turns channel 1 off)

both - Turns both channels on.

off - Turns both channels off.

Variable Selector

Overlay Mode

Specify how you want the image to appear on the screen. The value you enter is a command that is sent to the video overlay card installed on your system. When the video segment finishes playing, the overlay mode that was in effect prior to the V:Segment icon is automatically reinstated.

Note: Different video overlay cards support different commands. Refer to the documentation that came with your overlay card for more information on its capabilities.

Acceptable values are: **super**, **video**, **graph**, or a variable.

Drop-down List Box Items:

super - Displays video over graphics everywhere that the transparent color is used on the screen. By default, the transparent color is black. (If you want to specify a different transparent color, precede the V:Segment icon with a V:Overlay icon, specifying a different color in the Transparent Color text box.)

Use super overlay mode if you want to display graphics or text on the screen and then show video around the previously displayed material. For example, a Display icon displays a full screen graphic file that is completely black except for a title that appears in white text at the bottom of the screen. Next, a V:Segment icon plays a video segment. The segment appears everywhere on the screen except where the title (white text) appears. Note that if the transparent color does not appear anywhere on the screen, the video segment does not appear, although the audio can be heard.

video - Displays video only.

graph - Displays graphics only. When the segment plays, the audio portion of the recording is heard, but the video is not seen.

Variable Selector

Play From

Specify the starting frame number in the segment you want to play. When the V:Segment icon is executed, the video source seeks the specified frame number and plays when it is found.

Sometimes, it takes a videotape player a significant amount of time to find the starting frame. In this situation, it is recommended that at some point in the structure (prior to the V:Segment icon), you use a V:Still icon to pre-search for the starting frame. This prevents the pause that would occur when the V:Segment icon is looking for the correct frame and the video player is waiting to play the segment.

Acceptable values are: a frame number or a variable.

Drop-down List Box Items:

Video Editor
Variable Selector

Play To

Specify the ending frame number in the segment you want to play.

Acceptable values are: a frame number or a variable.

Drop-down List Box Items:

Video Editor
Variable Selector

Video Interrupt

Specify whether you want the user to be able to interrupt the video segment.

Acceptable values are: **no**, **yes**, or a variable.

Drop-down List Box Items:

no - The video segment plays from beginning to end and the user cannot pause, quit, or otherwise interrupt the video. Even the designated Break or Escape key is disabled while the segment plays.

yes - Causes a STOP button to appear in the lower right corner of the screen as the video plays. Depending on the available input device, the user can select the STOP button by clicking the mouse cursor on it, touching it, or pressing RETURN or the spacebar. When the STOP button is selected, a button and scrollbar appear that allow the user to control the video or quit viewing.

Variable Selector

Related Topic:

V:Segment Icon
Using Video Interrupt

Using Video Interrupt

When the user selects the STOP button during a video segment, the video pauses on the current frame, and the video interrupt buttons and slidebar are displayed:

Buttons

The way in which the user manipulates the buttons depends on the input device being used.

If a mouse or touch screen is being used, the user clicks on a button or touches it, respectively, to choose it. If a keyboard is being used, a button must be pre-selected before the user can choose it. When a button is pre-selected, it has a blinking underscore within a darkened boundary. By default, the Continue button is pre-selected. To pre-select a different button, the user presses Tab to advance to the next button. When the appropriate button is pre-selected, the user presses RETURN or the spacebar to choose it.

Continue - Starts the video segment from the current frame. (The current frame can be changed using the slide bar.)

Quit - Terminates the video segment. The icon below the V:Segment icon is executed.

Help - Accesses the current Help application.

Related Topic:

Creating a Help Application

Slide Bar

The slidebar allows the user to advance or rewind the video source. The way in which the user manipulates the slide bar depends on the input device being used.

If a mouse is used, each time the user clicks on the right arrow, the video steps forward one frame. Each time the user clicks on the left arrow, the video steps backward one frame. The user can drag the small white block to the left to rewind or to the right to fastforward. Dragging all the way to the left displays the first frame in the segment. Dragging all the way to the right displays the last frame in the segment.

If a touch screen is used, each time the user touches the right arrow, the video steps forward one frame. Each time the user touches the left arrow, the video steps backward one frame.

If a keyboard is used, each time the user presses the right arrow key, the video steps forward one frame. Each time the user presses the left arrow key, the video steps backward one frame.

V:Still Icon

The V:Still icon causes the video source to seek to a specified frame number. At a later point in the structure, a V:PlayTo or V:Player icon (with a playf command) causes the video source to play from this frame.

This icon can be used differently, depending on the type of video source you are using.

Videodisc Players

When the V:Still icon is executed the videodisc player begins seeking, and subsequent icons are executed immediately. Do not place the V:Still icon too close in the structure to the icon that causes the video to play. If the video plays before the frame is found, it will play from whatever frame is current and disregard the V:Still icon. In other words, if you know the videodisc player has a long way to seek before it finds the frame, or if your videodisc player is relatively slow, place the V:Still icon far enough ahead of the icon that plays the segment, so that the starting frame is found before the play command is given.

Videotape Players

Videotape players typically take a longer amount of time to seek to a specified frame. You can use the Wait Until Done? feature to prevent an icon that gives a play command (in a V:Player or V:PlayTo icon) from being executed until the appropriate frame is found.

Even if you set Wait Until Done? to yes, it is recommended that you place the V:Still icon that searches for a still frame, as far ahead of the icon that plays the video as is logically possible. This prevents the pause that will occur if the V:Still icon is still looking for the correct frame when a V:PlayTo or V:Player icon is waiting to run the video.

Another suggestion is that, whenever possible, plan the videotape so that the material it contains appears in the order in which it is used by the application. This also cuts down the amount of time it takes the V:Still icon to find the appropriate starting frame for a segment.

Content Editor Text Boxes:

Frame Number
Wait Until Done?

Frame Number

Specify the frame to which you want the video player to advance.

Acceptable values are: a frame number or a variable.

Drop-down List Box Item:

Variable Selector

Wait Until Done?

This text box is for videotape players only. If you are using a videotape player, you can use the Wait Until Done? text box to prevent a play command (in a V:Player or V:PlayTo icon) from being executed until the appropriate frame is found. This feature is useful because videotape players typically take a longer amount of time to seek to a specified frame.

Acceptable values are: **yes**, **no**, or a variable.

Drop-down List Box Items:

yes - Causes the player to begin seeking, and the subsequent icons are executed immediately, *unless* a V:PlayTo icon or a V:Player icon (with a playf command) is encountered. If either of these icons is encountered, IconAuthor *waits* until the appropriate frame is found before execution continues. Note that if any other V:Player icon is encountered (with a command other than playf), or if a second V:Still icon is encountered, it is disregarded, and execution flows to the next icon.

no - The V:Still icon executes the same way it executes when you are using a videodisc player. The player begins seeking, and the subsequent icons are executed immediately.

Variable Selector

Variable Icon

The Variable icon lets you manipulate variables. You can use a Variable icon to define a new variable, or to assign a new value to an existing variable.

Content Editor Text Boxes:

Variable Name
Assign Contents

Variable Name

Specify the name of the variable you want to define or to which you want to assign a new value.

The first character of every variable you create must be the @ symbol. Use upper and lower case letters, numbers, and the underscore symbol to name your variables. The maximum length of a variable name (including the @ symbol) is 19 characters.

Note: Do not create an application variable that begins with @_ This convention is reserved for system variables. Only reference a variable name that begins with @_ if you are deliberately changing the value in a system variable, an operation intended for advanced users.

Acceptable values are: a variable (a user variable, such as @COLOR, @NUM[1], a system variable such as @_GRAPHICS_PATH), or the expression Clear All.

Drop-down List Box Item:

Clear All - Clears all variables from memory.

Variable Selector

Assign Contents

Specify the value you want to assign to the variable in the Variable Name text box. You may enter several kinds of information in this text box. In general, information in the Assign Contents text box is a single value or an expression that manipulates or compares multiple values.

Acceptable values are:

- A real number (a number with a decimal point must have a digit to the left of the decimal point. For example, 0.9 and 0.23 are legitimate, but .9 and .23 are not.)
- 4 numbers that define an area on the screen (the first two are the x,y coordinates of the upper left corner of the area, the second two are the width and height of the area).
- A character string.
- A variable.
- A logical constant (.TRUE. or .FALSE., or more commonly, .T. or .F.).
- Expressions that contain numeric, relational, and/or logical operators.

Drop-down List Box Items:

Area Editor
Location Editor
Input Template Editor
Variable Selector

Related Topics:

Single Values in Assign Contents
Expressions in Assign Contents
Variable Icon

Single Values in Assign Contents

The single values that you can use are a number, a character string, a variable, or a logical constant (.TRUE. or .FALSE.). (.T. and .F. are legitimate abbreviations for .TRUE. and .FALSE.) Character strings must be surrounded by double quotation marks.

Related Topics:

Assign Contents
Variable Icon

Expressions in Assign Contents

You can use different kinds of operators to manipulate and/or compare different kinds of data in the Assign Contents text box. Use the following kinds of operators:

- numeric operators
- character operators
- relational operators
- logical operators

The kind of operators you use in the expression determine the kind of value that is stored in the Variable Name text box.

Related Topics:

Variable Icon
Assign Contents
Expressions in Assign Contents

Numeric Operators

You can specify an expression that uses one or more numeric operators. If you use this kind of expression, a number is returned to the variable in the Variable Name text box. Make sure to use a space before and after a numeric operator. For example, enter 5 + 5, not 5+5.

All numerical operations are done using real numbers. The following table describes the operations that are supported. Although the examples use numbers, variables can be used in place of any of these numbers. For example, in the case of addition, the Assign Contents text box could contain an expression like 2.3 + 2 or it could contain an expression like @HOURS + @MIN.

EXAMPLES:

Operator	Meaning	"Assign Contents"	"Variable Name"
+	addition	2.3 + 2	4.3
-	subtraction	6 - 1	5
*	multiplication	4 * 3	12
/	division	12 / 4	3
** or ^	exponent	2 ^ 3	8
SQRT(x)	square root	SQRT(4)	2
ABS(x)	absolute value	ABS(-52)	52
INT(x)	integer value	INT(2.3)	2
		INT(2.8)	2
		INT(2.854)	2
ROUND(x)	round off value	ROUND(2.3)	2
		ROUND(2.8)	3
		ROUND(2.3333)	2
SIN(x)	sine value	SIN(30)	0.5
COS(x)	cosine value	COS(60)	0.5
TAN(x)	tangent value	TAN(0)	0.0
LOG(x)	natural logarithm	LOG(1.0)	0.0
LOG10(x)	logarithm base 10	LOG10(10)	1.0
+	positive	+2.3	2.3
-	negative	-3	-3

You can use these numerical operations individually within the Assign Contents text box, or you can combine them. For example, you can specify 2 * 3 + 4 (which causes "10" to be assigned to the variable in the Variable Name text box). Normal mathematical and left-to-right precedence rules apply, unless parentheses are used. For example, while 2 * 3 + 4 = 10, 2 * (3 + 4) = 14.

Related Topics:

Variable Icon
Assign Contents
Expressions in Assign Contents

Character Operators

You can specify an expression that uses one of the following character operators to manipulate character strings:

&
$
EXTRACT(start,len,str)
UPPER(str)
LOWER(str)

The character operator "&" concatenates (adds) one string to another and places the result in the variable in the Variable Name text box. To include a space between two strings being concatenated, add a space to the end of the first string.

The character operator "$" evaluates whether the first argument is included within the second and returns a value of .T. or .F. to the variable in the Variable Name text box.

The character operator EXTRACT() extracts a portion of a string and places it in the variable in the Variable Name text box. The syntax for this operator is EXTRACT(start,len,str), where:

start is the position of the first character you want to extract

len is the length of the portion you want to extract

str is the name of the variable being manipulated

The character operator UPPER() converts a string specified within the parentheses to uppercase. The character operator LOWER() converts the string to lowercase. If you use one of these case conversion operators alone in the Assign Contents text box, it changes the value of the variable in the Variable Name text box.

Note: If you use a case conversion operator as part of a comparison in the Assign Contents text box, the value being converted is only converted for the comparison.

EXAMPLES:

Operator	**Meaning**	**"Assign Contents"**	**"Variable Name"**
&	concatenate	"Hello " & "Joe"	"Hello Joe"
$	inclusive	"A" $ "ABC"	.T.
EXTRACT()	extract part of a string	EXTRACT(1,4,@DATE)	"1991"
UPPER()	convert to uppercase	"abc"	"ABC"
LOWER()	convert to lowercase	"ABC"	"abc"

Related Topics:

Variable Icon
Assign Contents
Expressions in Assign Contents

Relational Operators

Relational Operators and Numbers

You can specify an expression that uses a relational operator to compare numbers. When you do a comparison of numbers in the Assign Contents text box, a logical result .T. (for true) or .F. (for false) is stored in the variable in the Variable Name text box.

Use a space before and after a comparison operator. For example, enter @NUMBER > 5, not @NUMBER>5.

The items you are comparing can contain mathematical operations. Normal mathematical, and left to right precedence rules apply, unless parentheses are used.

EXAMPLES:

Operator	Meaning	"Assign Contents"	"Variable Name"
<	less than	2 < 4	.T.
<=	less than or equal to	2 <= 1	.F.
=	equal to	2 * 2 = 4	.T.
>	greater than	2 > 4	.F.
>=	greater than or equal to	3 >= 3	.T.
<> or #	not equal to	2 * 3 <> 4	.T.

Relational Operators and Character Strings

You can specify an expression that uses a relational operator to compare character strings. When you do a comparison of character strings in the Assign Contents text box, a logical result .T. (for true) or .F. (for false) is stored in the variable in the Variable Name text box.

Use a space before and after a comparison operator. For example, enter "JANE" = "Jane", not "JANE"="Jane".

Character strings are identified with quotation marks. If a character string comparison is being done on strings within variables, the STRING operation must be done on one of the variables.

When character strings are compared, they are evaluated in terms of the position of each character in the standard ASCII table. The first character in one string is compared to the first character in the other string. If the two are equal, the second characters are compared and so on.

EXAMPLES:

Operator	Meaning	"Assign Contents"	"Variable Name"
<	less than	"A" < "a"	.T.
<=	less than or equal to	"ABC" <= "ABCD	.T.
=	equal to	"ABC" = "ABC"	.T.
>	greater than	"ABCD" = "abcd"	.F.
>=	greater than or equal to	"ABC" >= "DEF"	.F.
<> or #	not equal to	"ABC" <> "abc"	.T.
$	contains	"a" $ "abc"	.T.

Logical Operators

You can specify an expression that uses one or more logical operators (.AND., .OR., and .NOT.) to compare expressions that contain numbers, character strings, and the logical constants .T. and .F. When you use logical operators to evaluate expressions in the Assign Contents text box, a Logical result .T. (for true) or .F. (for false) is stored in the variable in the Variable Name text box.

Use a space before and after a logical operator. For example, enter @WIDTH > 80 .AND. @HEIGHT > 60, not @WIDTH>80.AND.@HEIGHT>60.

.AND. asserts that both expressions are true.

.OR. asserts that at least one of the comparisons is true.

.NOT. asserts that the negative of the comparison is true.

WaveAudio Icon

The WaveAudio icon is a composite that allows you to play wave audio files if you are using the Multimedia Extensions software and an audio card such as the SoundBlaster (Creative Labs) or the Pro Audio Spectrum (Media Vision).

Suggested Use:

- Play a wave audio file to provide feedback to a user, such as "Please press any key to continue."

Three MCI icons form the backbone of the WaveAudio composite. Although you must be familiar with the MCI syntax in order to fully take advantage of the MCI feature, the WaveAudio composite already contains some values (that require minimal editing) so that you can quickly start playing wave files as part of your IconAuthor applications. Once you become familiar with the MCI syntax you can customize and vary the commands. For information on the MCI command syntax open the help file called MCISTRWH.HLP.

The composite contains a mini-structure of icons:

1. MCI icon: Contains the command **open c:\iauthor\audio\filename.wav type waveaudio alias sound**, where: **open** initializes the device and **c:\iauthor\audio\filename.wav** represents the path and filename of the wave audio file to be played. You must change this information so that it indicates the specific path and filename you are playing. The **type waveaudio** parameter is the type of device and **alias sound** specifies the name "sound" as an alternate name for the waveaudio device type.
2. MCI icon: contains the command **play sound** which starts the wave audio file playing.
3. Input icon: Causes execution flow to stop at this point and wait for the user to provide input. This icon specifies that the entire screen is input selectable. That means that the user can click anywhere or press any key to cause execution flow to continue.
4. MCI icon: When the user clicks, execution flows to the third MCI icon which contains the command **close sound** to suspend playback and relinquish access to the device.

Hint: If you want to use this composite to play a wave audio file while some other activity is occurring, replace the Input icon with one or more alternative icons. For example, if you use a Display icon (in place of the Input icon) to run an animation script, the audio will play, the animation will run, and when the animation completes, the audio file will be closed.

Content Editor Text Box:

The lead icon in the WaveAudio composite is labeled "WaveAudio" and contains only one text box "Composite Name". Enter a different name in this text box to customize the name of the composite.

Related Topics:

MCI Icon
Input Icon

Window Icon

The Window icon lets you run your IconAuthor application in a window. When you run IconAuthor in a window, you give the user the ability to run other applications simultaneously. For example, if your application is teaching a user how to use another application, the tutorial created with IconAuthor can run in a window. It can instruct the user on the application running in another window.

You can specify the window size, window title, the background color of the window, and whether the window is resizable.

Use of the Window icon is optional. If you do not use a Window icon in your application, IconAuthor uses default settings to display the application. The default settings cause the application to appear full screen with a white background. You can change the default settings by choosing Default Window Setup... from the Run menu of the IconAuthor window. The Default Window Setup dialog box appears and lets you specify settings for the size, title, re-sizability, and color of a window.

Note: In many situations, a Window icon is one of the first icons in your structure. If you choose to run the application from a selected icon (and the Window icon is not included in the icons that are executed), the default window settings in the Default Window Setup dialog box are used.

You can include one or more Window icons in your application, although only one IconAuthor window can exist at one time.

The Window Object

Technically, any application that displays information to the screen runs in a window. The window in which the application runs is actually an object. Like the other objects available in IconAuthor, the Window object can be manipulated at runtime. For example, you use an ObjSet icon to change the background color of the window by re-setting its ColorBackground property.

Content Editor Text Boxes:

Window Size
Window Title
Window Re-size
Window Color

Related Topics:

Using the Window Object
Changing Window Properties
Window Properties

Window Size

Specify the size of the **client area** of the window. The client area is the area of the window exclusive of the title bar and window borders.

You can make the application appear full screen or you can specify dimensions so that the application appears in a window. If you specify a size that is the same as the screen resolution (for example, 640 x 480) the application appears full screen. If you specify a size that is smaller than the screen resolution in either the X or Y dimension, the window is movable. If you specify a size that is larger than the screen resolution in either the X or Y dimensions, the window is not movable, does not have a title bar, and is centered on the screen.

Acceptable values are: **full screen**, 4 numbers (separated by commas) that define a window's area on the screen (the first two numbers are the x,y coordinates of the upper left corner of the window, the second two numbers are the width and height of the window), or a variable.

Drop-down List Box Items:

full screen - Causes the IconAuthor application to appear full screen on the monitor. The application has no title bar and is not resizable.

Area Editor

Variable Selector

Window Title

Specify the text you want to appear in the title bar of the window.

Note: If you specified full screen as the window size, do not specify a window title. If you do specify a title, it is ignored.

Acceptable values are: text that represents a title, or a variable. The number of characters you can enter depends on the size of the window.

Drop-down List Box Item:

Variable Selector

Window Re-Size

Specify whether you want the window to be resizable.

Note: If you specified full screen as the window size, do not specify a value in this text box. Any value you specify is ignored.

Acceptable values are: **no**, **yes**, or a variable.

Drop-down List Box Items:

no - The window is movable, but not resizable. It cannot be minimized to an icon.

yes - The window is movable, resizable, and it can be minimized to an icon. When the user minimizes the application, execution halts. Execution resumes when the application is restored or maximized.

Variable Selector

Window Color

Specify the background color of the window.

Acceptable values: are a color name, an **RGB** value, or a variable.

Drop-down List Box Items:

assorted colors - Frequently used colors such as white, black, red, etc.

Color Editor

Variable Selector

Write Icon

The Write icon dynamically displays text or the value stored in a variable on the screen. The Write icon can display numbers or strings, however, only one line of characters can be displayed by each Write icon. The number of characters that you can display in one line depends on your system, and the font type and font size you are using.

Suggested Uses:

- Display a brief question or message on the screen.
- Display the content of a variable on the screen.

The Write icon is useful for debugging applications because you can display the contents of variables on the screen, during execution. For example, you may create an application that contains an indexed loop. You intend for the loop to execute three times, however, it appears to execute only two times. To test whether the index variable is incrementing properly, place a Write icon under the Loop Start icon. The Text to Display text box is @COUNT. Next, place a Pause icon below the Write icon to display the current value of @COUNT at the location specified.

To set the font type, font size, and/or color that are used to generate the characters on the screen, precede the Write icon with a Font icon and/or a Color icon. If you do not use these icons, the system default font, size, and color are in effect.

Note: When you use the Font and Color icons, the new settings are in effect until you use subsequent Font or Color icons.

Content Editor Text Boxes:

Text To Display
Location

Related Topics:

Font Icon

Text To Display

Specify the text or variable you want to display. You cannot use one Write icon to display both text and the contents of a variable. If you are displaying text, *do not* enclose the text in quotation marks unless you want the text to appear on the screen within quotation marks.

Acceptable values are: a text string or a variable.

Drop-down List Box Item:

Variable Selector

Related Topic:

Write Icon
Font Icon

Location

Specify the location where you want the text or the contents of the variable to appear on the screen. Indicate the location by specifying the coordinates of a point on the screen. This point is the upper left corner of the area occupied by the characters displayed on the screen. This means that if you select a point, the text is displayed below and to the right of that point.

Acceptable values are: a pair of coordinates (separated by commas) that define the upper left corner of the area where text will appear on the screen, or a variable.

Drop-down List Box Items:

Location Editor
Variable Selector

RGB Values

An RGB value is made up of three numbers separated by commas, such as 0,255,127. Each number represents the level of intensity of red, green, and blue, respectively, that make up the resulting color. Each intensity level value can range from 0 to 255. 0,0,0 is black and 255,255,255 is white.

Subroutine

A subroutine is a structure of icons (contained in its own .IW file) that performs a specific function. The subroutine is not used on its own as a standalone application, but can be called as often as necessary by other IconAuthor main or sub-application. A subroutine can be used repeatedly by one application, and it can also be called by several different applications.

Icon Library

The Icon Library is the scrollable area on the left side of the IconAuthor window that contains the icons you use to build a structure. Initially, the library contains folders each of which represents a category of icons. You can open and close one folder or all of the folders to bring icons into and out of view. When the icons in a folder are visible, you can build them into your application structure.

Ribbon Bar

The graphic bar across the top of the work area. Press on-screen buttons to quickly perform some of the more frequently used IconAuthor tasks.

Status Bar

The message area at the bottom left corner of the IconAuthor window. The message in the status bar indicates the status of the current item on which the mouse cursor is positioned.

Hint: Position the mouse on an icon in the structure. The content for that icon is described in the status bar.

Structure

The flowchart (logical sequence) of icons you create in the work area.

Application Variables

Application variables (also called user variables) are so named because you create them as part of your application. Like all IconAuthor variables, application variables are structures designed to hold values that are assigned while your application is running. When you use a variable (instead of a fixed value) in a Content Editor text box, the icon can perform differently depending on the current value stored in the variable.

Composite Icon

A composite icon is one icon in the library that actually represents a mini-structure of several icons. Composites provide a way for saving groups of icons that represent a portion of a structure that is commonly used in many applications or repeatedly used in one. IconAuthor contains some composite icons in the initial library. You can also create an save your own composites.

Direct Value

When you click on a direct value in a Content Editor drop-down list box, such as a number or color, the drop-down list box is closed and the value is automatically entered in the corresponding text box. If the text box already contained a value, that value is overwritten by the new selection.

Indirect Value

When you click on an indirect value in a Content Editor drop-down list box, you access one of several tools to help you locate or create a value for the corresponding text box. In other words, when you click on an indirect value, the item is *not* automatically entered in the corresponding text box, but its tool is opened.

Single Value Application Variables

Single value application variables contain only one piece of information, such as a number or a filename.

Loop

A loop is a group of icons in your structure that is executed more than once because execution flows round and round in a circular fashion. Rather than building the same logic into your structure repeatedly, a loop is efficient because it uses the *same* logic, or, the same portion of the structure over and over again.

Variable File

A variable file is an ASCII text file. The file contains any number of application variables (single value and/or indexed) and can be created with a text editor such as Notepad, or using a text window within IconAuthor. When you name a variable file, use a .VAR extension. Include a carriage return at the end of the last line in a variable file.

Document Window

A document window is a window that you can open within the IconAuthor work area. You can open a document window that contains an application file or a text file. (In IconAuthor these are referred to as an application window and a text window, respectively). You can also open document windows in IconAuthor that contain a SmartObject page or a graphic file. Document windows can be moved, resized, minimized, etc. You can open several document windows in the work area at once.

SmartObject Editor

When you select the SmartObject Editor item, the editor appears and can be used to view, create, and edit one or more SmartObject files. When you close the SmartObject Editor, the name of the last file and page with which you were working is returned to the appropriate Content Editor text boxes.

Animation Editor

When you select the Animation Editor item, the animation editor, IconAnimate, appears and can be used to view, create, and edit one or more animation files. When you close IconAnimate, the name of the last file with which you were working is returned to the appropriate Content Editor text box.

Video Editor

The Video Editor lets you view a videodisc or video tape through IconAuthor. When you access the Video Editor through a Content Editor, you can use the editor to select a starting and/or ending frame of a segment. When you close the Video Editor you have the option of returning the selected frame number to the appropriate Content Editor text box.

Area Editor

The Area Editor is accessed from one of several Content Editor drop-down list boxes. Use the Area Editor to select an area on the screen. When you close the Area Editor, the coordinates of the area that you selected are returned to one or more appropriate text boxes.

When you access the Area Editor the screen background becomes black and the Area Editor appears.

Before you select an area, you can optionally display a bitmap graphic, SmartObject page, or video frame on the background to simulate how the screen will look at runtime. If you choose to display a graphic, SmartObject page, or video frame, the appropriate check boxes are turned on. The next time you access the Area Editor, those display mechanisms are still set to on. To stop displaying a graphic, etc., de-select the appropriate check box.

To select an area:

1. Position the pointer at the location that you want to be the upper left corner of the box-shaped area you are selecting.
2. Press the mouse button and drag down and to the right to draw a box-shaped area.

   If you decide to stop selecting an area, click on the right mouse button.
3. When you are satisfied with the size of the area, release the mouse button.

   The numbers in the Coordinates text box are fixed and reflect the upper left corner, width, and height of the box. If you want to select a different area, repeat steps 1 through 3 as many times as necessary. Each time you begin selecting an area, the previously selected area disappears.
4. Click OK.

   Key coordinates of the area you selected are returned to one or more appropriate Content Editor text boxes.

Related Topics:

Displaying a Bitmap in the Editor
Displaying a SmartObject Page in the Editor
Displaying a Video Frame in the Editor

Displaying a Bitmap Graphic in the Editor

1. Select Display Graphics.
   A File Open dialog box appears.
2. Double click on the file you want to display.
   The file is displayed with its upper left corner located at the upper left corner of the screen.

Displaying a SmartObject Page in the Editor

1. Select Display Text.
   A File Open dialog box appears.
2. Double click on the file you want to display.
   A SmartObject page Selection dialog box appears.
3. Double click on the page you want to display.
   The page is displayed with its upper left corner located at the upper left corner of the screen.

Displaying a Video Frame in the Editor

To display the current video frame:

- Select Video Overlay.

  The current video frame is displayed.

Note: The Video Editor is only available if you have installed and configured the necessary video hardware and software.

To use the Video Editor to display a frame:

- Choose Video Editor.

  The Video Editor appears and you can use the Video Editor control panel to display a specific frame. Close the Video Editor and that frame is displayed while you use the Area Editor.

Note: The Video Editor is only available if you have installed and configured the necessary video hardware and software.

Location Editor

The Location Editor is accessed from one of several Content Editor drop-down list boxes. Use the Location Editor to select a point on the screen. When you close the Location Editor, the coordinates of the point that you selected are returned to the appropriate text box.

When you access the Location Editor the screen background becomes black and the Location Editor appears.

Before you select an area, you can optionally display a bitmap graphic, SmartObject page, or video frame on the background to simulate how the screen will look at runtime.

If you choose to display a graphic, SmartObject page, or video frame, the appropriate check boxes are turned on. The next time you access the Location Editor, those display mechanisms are still set to on. To stop displaying a graphic, etc., de-select the appropriate check box.

To select a location:

1. Move the mouse cursor to the point you want to select.
2. Click on the location that you want to select.

   A small rectangle appears at the selected point. The coordinates of the point appear in the Location Editor's Coordinates text box. If you want to select a different location, repeat steps 1 and 2 as many times as necessary. Each time you select a location, the coordinates of the new location appear in the Coordinates text box.
3. Click OK.

   The coordinates of the location you selected are returned to the appropriate Content Editor text box.

Related Topics:

Displaying a Bitmap in the Editor
Displaying a SmartObject Page in the Editor
Displaying a Video Frame in the Editor

Input Template Editor

Typically, you access the Input Template Editor from a drop-down list box of the InputMenu icon Content Editor. You can also access this editor through the Variable icon. Use the editor to create a template of selectable areas on the screen. Optionally, you can save this information to a template file so that you can easily use the same selectable areas for another InputMenu or Variable icon. When you close the editor the coordinates of the areas you specified are returned to the appropriate text box.

When you access the Input Template Editor the screen background becomes black and the Input Template Editor appears.

Before you select an area, you can optionally display a bitmap graphic, SmartObject page, or video frame on the background to simulate how the screen will look at runtime.

To select input template areas:

1. Position the pointer at the location that you want to be the upper left corner of a selectable area. (If the Input Template Editor obstructs the point you want to select you can drag it out of the way.)
2. Press the mouse button and drag down and to the right to draw a box-shaped area.

   When you press the mouse button four numbers separated by commas appear in the editor's Coordinates area. The first two numbers are the coordinates of the upper left corner of the area you are selecting. The second two numbers are 0,0, and represent the current width and height of the area. As you drag down and to the right, the second two numbers change to reflect the current width and height.

   If you want to cancel your selection, click on the right mouse button.
3. When you are satisfied with the size of the area, release the mouse button and the numbers are recorded as the first template.

   If you want to create more areas, repeat steps 1 through 3 as many times as necessary. Each time you define a new area, the template number is incremented by one. The number recorded for the template corresponds to the branch number in the InputMenu composite icon.

   The numbers in the Coordinates text box are fixed. If you want to delete the area, click on it and choose Delete.
4. Click OK.

   The coordinates of the selectable areas are returned to the appropriate Content Editor text box.

To resize a template area:

1. Click on the template area you wish to resize. Eight resizing blocks are displayed around the template area.
2. Position the cursor over the resizing block. It will become a double headed arrow.
3. Press the left mouse button and drag the cursor.

To move a template area:

1. Click on the template you wish to move to select it.
2. With the cursor positioned anywhere within the area, press the left mouse button and drag the mouse pointer to move the template area to the new location.

Related Topics:

Working with Template Files
Displaying a Bitmap in the Editor
Displaying a SmartObject Page in the Editor
Displaying a Video Frame in the Editor

Working with Template Files

IconAuthor also gives you the option of saving templates in files, so that you can easily use them in other InputMenu and Variable icons. If you decide to save a template as a file, name it with a .TEM extension and store it in your INPUT subdirectory.

To save a new template file:

1. Choose Save As... from the File menu of the editor.

   A File Save As dialog box appears.
2. Type the name of the new template file in the Filename text box.
3. Choose OK.

   If you want to begin creating another new template file, choose New from the File menu of the editor, specify the selectable areas, and choose Save again.

To save changes to a template file:

- Choose Save from the File menu of the editor.

  The template file is automatically saved.

To open an existing template file:

1. Choose Open... from the File menu of the editor.

   The File Open dialog box appears.
2. Select a file from the Files list box.
3. Choose OK.

   The file you chose is opened and the selectable areas it specifies are visible in the editor window. To return the coordinates of these areas to the Content Editor, choose OK.

Displaying a Bitmap Graphic in the Input Template Editor

1. Choose Graphics File... from the Show menu.
   A File Open dialog box appears.
2. Double click on the file you want to display.
   The file is displayed with its upper left corner located at the upper left corner of the screen.

Displaying a SmartObject Page in the Input Template Editor

1. Choose Text File... from the Show menu.
   A File Open dialog box appears.
2. Double click on the file you want to display.
   A SmartObject Page Selection dialog box appears.
3. Double click on the page you want to display.
   The page is displayed with its upper left corner located at the upper left corner of the screen.

Displaying a Video Frame in the Input Template Editor

Note: Video is only available if you have installed and configured the necessary video hardware and software.

To display the current video frame:

- Choose Overlay from the Show menu.

  The current video frame is displayed.

To use the Video Editor to display a frame:

- Choose Video Editor... from the Show menu.

  The Video Editor appears and you can use the Video Editor control panel to display a specific frame. Close the Video Editor and that frame is displayed while you use the Input Template Editor. For information on how to use the Video Editor, see Chapter 12.

Color Dialog Box

The Color dialog box is accessed from one of several Content Editor drop-down list boxes. Use the Color dialog box to select a color or create a custom color. When you access the Color dialog box through a Content Editor, work with it, and close it, the last selected color is returned to the appropriate text box in the form of a keyword or an **RGB** (red green blue) value.

Related Topics:

Selecting a Basic Color
Creating a Custom Color

Selecting a Basic Color

To choose a basic color:

1. Select one of the 48 color cells in the Basic Colors palette.
2. Choose OK to accept the change or choose Cancel to dismiss the dialog box without accepting changes.

Creating a Custom Color

To create a custom color:

1. Click on the Define Custom Colors... button in the Colors dialog box.

   The Color is extended to show the custom color selector.

2. Click on an empty cell in the Custom Colors area.

3. Drag the color refiner cursor to the area of the color refiner box that shows the color you want to use. Then drag the arrow next to the vertical luminosity bar up or down to adjust the luminosity.

   As you change the color, the new color is displayed on the left side of the Color/Solid box. The right side of the box displays the solid color closest to your choice. If you want to select the solid color, double-click on the right side of the box.

4. When you are satisfied with the color, click on the Add to Custom Colors button.

   The color is added to the selected cell.

5. Define any other colors you want to add to the palette and choose OK to accept the changes and close the Colors dialog box.

Variable Selector

Access the Variable Selector from any Content Editor drop-down list box. Use the Variable Selector to select an application or system variable. When you close the Variable Selector, the variable you selected is returned to the appropriate text box. Using the Variable Selector reduces the chance of typing errors.

System Variables automatically appear in the Variable Selector regardless of whether they have values. Application variables only appear in the Variable Selector after an application has been run.

To use the Variable Selector:

1. Choose Variable Selector from a drop-down list box of a Content Editor.

   The Variable Selector appears.

2. Select a variable from the list.

   If the application has been run and contains application variables, those variables appear in alphabetical order at the beginning of the list. System variables appear at the end of the list.

3. Choose OK.

   The Variable Selector is closed and the selected variable is returned to the appropriate text box.

Directory

The Directory item lets you search for the name of a file to enter in a Content Editor text box. To access the Directory, select that item from the list box. A Directory file selection dialog box appears.

The Directory dialog box varies depending on the type of file for which you are searching. For example, when you use the Directory item from the Display icon and have indicated you are searching for a bitmap graphic file type, the dialog box comes up with the *.BMP, *.RLE, and *.PCX filters and searches your graphic file subdirectory for matching files.

Select the filename you want and choose OK. The Directory dialog box is closed and the filename is entered in the Filename text box.

Font Dialog Box

To change font information:

1. Select the font you want to use from the Font list box.

   The fonts that are available are those that are currently loaded in Windows. A sample of the font that you select appears in the Sample area.

2. As necessary, adjust the Font Style and Size.

   The available options vary depending on the currently selected font.

3. As necessary, click on the Strikeout and Underline options in the Effects area.
4. Use the drop-down list in the Color area to change the color of text.
5. When you are satisfied with the font as it appears in the Sample area, choose OK to close the Font dialog box.

Object Name Selector

Use the Object Name Selector dialog box to easily locate an Object Name to return to the Content Editor. The Object Name Selector dialog box gives you a way of viewing ObjectNames without running the SmartObject Editor and opening the correct file and page.

1. Choose Object Name Selector from the Name drop-down list box of an object icon. It is also available from the If icon's Condition 2 text box.
2. Select the SmartObject file that you want to access.

   The File drop-down list shows a history of the last five files selected. To choose a different file, click on the Browse... button. When the Open dialog box appears, locate and double-click on the desired file.
3. Use the Page drop-down list to select a Page in the specified file.

   The Page drop-down list box shows all the pages in the file.
4. Use the Name drop-down list to select an ObjectName.

   The Name list shows all the objects on the selected page. If an object is unnamed, it is represented by the expression "[empty]."
5. Choose OK to close the ObjectName Selector.

After you use the Object Name Selector to find an ObjectName for an ObjSet or ObjGet icon, the Property drop-down list conveniently displays the properties specific to the selected object. In the ObjSet icon, the Value drop-down list also displays the potential values that are appropriate for the selected property. The value to which the property was initially set in the SmartObject Editor is preceded by an asterisk ("*").

Object Event Selector

Use the Object Event Selector to select an event that you want to use in the If icon's Condition 2 text box. This tool is convenient when an If icon is testing whether the value in @_Object_Event is equal to a particular event.

To use the selector:

1. Choose the appropriate object class from the drop-down list.

   All possible events for the selected class appear in the list box.
2. Choose an event.
3. Click on OK to close the selector.

Icons

Icons are the building blocks of an IconAuthor application. Each icon is a small picture that represents a function that can be performed. As an example, the icon to the right is a Display icon that allows your application to display a file (such as a graphic) on the screen.

Related Topics:

Icon Library
Building with Icons
Defining How Icons Perform

Icons:

Beep Icon
Box Icon
Branches Icon
CD-Audio Icon
Circle Icon
Clear Icon
Color Icon
DDE Icon
DllCall Icon
Dll Link Icon
Database Icon
Date&Time Icon
Display Icon
Ellipse Icon
Exit Icon
Font Icon
Help Icon
If Icon
Input Icon
InputMenu Icon
Line Icon
LoadVar Icon
Loop Icon
LoopIndex Icon
MCI Icon
Menu Icon
MIDI Icon
Module Icon
MsgBox Icon
Note Icon
ObjDelete Icon
ObjEvent Icon
ObjGet Icon
ObjMenu Icon
ObjQueue Icon
ObjSet Icon
Parse Icon
Pause Icon
Print Icon
Program Icon
RS-232 Icon
Random Icon
SaveVar Icon
Shuffle Icon

Snapshot Icon
Startup Icon
SubApp Icon
SubAssign Icon
Subroutine Icon
Text Icon
V:Audio Icon
V: Frame #? Icon
V: Overlay Icon
V:PlayTo Icon
V:Player Icon
V:Segment Icon
V:Still Icon
Variable Icon
WaveAudio Icon
Window Icon
Write Icon

Working with Objects

Objects are as essential to applications as icons. While icons make up the structure of your application, controlling which action takes place in what order, objects also play a key role in determining the appearance and performance of your application. The following objects are available for use in your applications:

Audio
Button
Combo Box
Database
Graphic
IconAnimate
Keyboard
List Box
Menu
Movie
OLE
Text
Timer
Transparent
Variable

You use IconAuthors **SmartObject Editor** to create a file that contains these kinds of objects. In the editor you position the objects exactly where you want them and you customize their appearance so that they look just right. For example, first you draw a Button, then you move it and resize it, then you can set its label so that is says OK.

Two additional live objects are automatically available in every application. Unlike the other objects, you do not create these objects in the SmartObject Editor.

System
Window

Defining how objects look and perform is called **setting properties**. Every object has properties (characteristics) that you can set. For example, in order to turn a Button object into an OK button, you set the objects Label property to OK.

All the objects are **live** by default. When an object is live, the user can interact with it at runtime. For example, a Button object is live and therefore a user can click on it. Also, when an object is live, your application can manipulate it. For example, if a Button object is live you can use an icon to change its Label property at runtime so that it changes from a Play button to a Pause button.

You have the option of making a small number of objects **static** instead of live. However, once static objects are displayed, they behave as if they are part of the background. Your application cannot set properties of static objects at runtime.

Building Structures

To begin building a structure you must first use a command in the File menu to open an application window. You can start a new application (using the New... command) or open an existing one (using the Open... command). Once the application window is open, you begin building by:

1. Finding the appropriate icon in the Icon Library.
2. Dragging the icon to the desired position in the structure.
3. Dropping the icon into position.

Related Topics:

Finding Icons in the Icon Library
Jumping to an Icon in the Library
Dragging and Dropping Icons

Finding Icons in the Icon Library

When you first open IconAuthor, the icon library displays the contents of the IAUTHOR.LIB file. To see more of the library contents, use the scroll bar to scroll up or down. The IAUTHOR.LIB library file contains the most frequently used IconAuthor icons arranged alphabetically. If you prefer to have all the IconAuthor icons available, load the library file to IAFULL.LIB.

Jumping to an Icon in the Library

Instead of scrolling through the library to find an icon, you can use a special feature to automatically jump to a particular alphabetical portion of the library.

To jump to an icon in the library:

1. Click on any icon in the library.
2. On the keyboard, press the first letter of the icon you want to find.

   The library jumps to and highlights the next icon that begins with the chosen letter. If you press the letter again, IconAuthor finds the next icon that begins with that letter. When it reaches the end of the library, IconAuthor loops back to search from the beginning of the library.

Dragging and Dropping Icons

The technique you use to move icons from the library into a structure is called **drag and drop**. When you build an icon from the library into a structure you actually use a *copy* of that icon. The original icon remains unchanged and available for use in the Icon Library.

To drag and drop an icon into a structure:

1. Position the arrow-shaped mouse pointer over an icon in the library.
2. Press and hold the left mouse button.

   The icon is selected and appears indented as though it is a button that has been pushed in.
3. Continuing to hold the mouse button down, drag the icon to the point in the structure where you want to build it.

   As soon as you drag, the mouse cursor assumes the shape of the selected icon. The original icon in the library remains unchanged and you are now dragging a copy of the icon into the application window. If this is the first icon you are building, drag the icon until it is just below the Start icon. If there are already other icons in the structure, you can drag the icon to a point just below one of the icons or between two icons.

   Most icons can only be built just below or just above another icon. However, some icons, such as the If icon can also be built just to the right of an icon in the structure.
4. Drop the icon into the structure by releasing the mouse button.

   Icons can only be dropped in the following valid positions: the structure, the Clipboard, and the Trash Can. If the icon is dragged to an invalid position the cursor changes from the icon shape to a red circle with a diagonal bar through it. If you drop the icon in an invalid position, the drop is aborted. When you re-enter valid areas the cursor changes back to the icon cursor.

   If you drop the icon into the structure in a valid position, the icon becomes part of the structure and the cursor reverts to the shape of the hand holding the pen.

Note: Most often, when you build an icon from the Icon Library into a structure, the icon, as it appears in the library, is added to the structure in the window. The exception to this rule is the composite icon. A composite icon is one icon in the library that actually represents a mini-structure of several icons in the structure.

You can build as many icons as you like into the structure. At any time, you can move on to other tasks such as adding content to the icons you have built into your application or editing the application structure.

Composite Icons

A composite icon is one icon in the library that actually represents a mini-structure of several icons. Composites provide a way for saving groups of icons that represent a portion of a structure that is commonly used in many applications or repeatedly used in one. IconAuthor contains some composite icons in the initial library. You can also create and save your own composites which become part of the icon library and can be used in multiple applications.

When you select a composite from the library and build it into your structure, the icon structure of the composite will appear in your structure, allowing you to add content to each of the icons in the composite.

Icons in the library that are composites are:

Branches Icon
Loop Icon
LoopIndex Icon
ObjMenu Icon

As an example, when you find the Loop icon (a composite) in the library and build it into the structure, a mini-structure of 3 icons appears. The first icon in a composite (in this case, the one labeled "Loop") is called the **lead** icon.

Adding Content

After the structure is built, you **add content** to each of the icons in the structure. Through a series of dialog boxes, you specify what the application will do when it is executed. Every icon has its own dialog box associated with it called a **Content Editor**. For example, there is a Box icon Content Editor, a Pause icon Content Editor, and so on.

Related Topics:

Content Editors
Icon Colors
Naming Icons
Entering Values
Accessing Editors or other Dialog Boxes

Editing Applications

As you create your application, you can run and check it at any time. The process of detecting and fixing errors in the structure or content is called debugging. You can debug your application by simply running it and checking each piece of functionality or you can run the application in conjunction with IconAuthors visual debugging tool called IAScope. IAScope lets you run the application and view the content of each icon as it executes. If something unexpected occurs during execution, you can take a look at the IAScope viewer to see which icon is the source of the problem.

As you check your application, you can edit either the structure, the content, or both. When you edit the structure you perform tasks like copying icons and inserting them elsewhere in the structure, and deleting icons from the structure. When you edit the content of the application you re-open an icon's Content Editor and change the values. You can change only the values in the dialog box, or the underlying content file (such as a graphic or animation) that the Content Editor invokes.

Related Topics:

Editing Structures
Running an Application
General Execution Rules
Debugging

The IconAuthor Window

The IconAuthor window contains the following the major components:

Title Bar

The IconAuthor title bar contains the name of the application, "IconAuthor."

Menu Bar

The IconAuthor menu bar has seven pull-down menus. The menus allow you to create, edit, and manage your applications, tailor the appearance of your IconAuthor window, and access online Help.

Ribbon Bar

The ribbon bar is a graphic bar across the top of the IconAuthor window. You can click on these buttons to quickly perform some of the more frequently used IconAuthor tasks. All of the functions in the ribbon bar are also available through the commands in the menu bar.

Work Area

The work area is where you open the window in which you create applications. Initially, the work area contains a graphic image of the IconAuthor logo. As soon as you pull down any menu in the menu bar the logo disappears and the work area is cleared.

Icon Library

The Icon Library (also referred to as simply the "library") contains the icons that you use to build a structure within an application window. The most frequently used icons appear in the library by default, and are contained in a library file called IAUTHOR.LIB. Additional icons are available in an alternative library file called IAFULL.LIB.

Status bar

The status bar displays information about the icon on which the cursor is currently positioned. For example, when the cursor is over the Start icon the status bar shows the message 1,1:Start. This indicates that the icon is in the first row of the first column of icons and it's name is Start. When you add content to an icon, the data you provide also appears in the status bar.

Starting an Application

The first time you begin work on an application you must open a new application window.

To open a new application window:

1. Choose New... from the File menu.

   The New dialog box appears with the Application Window option selected.

2. Choose OK to open a new application window.

   A blank, untitled application window appears.

The new application window automatically contains a Start icon in the top left corner. The Start icon is not in the Icon Library because it is built into every structure. It cannot be moved, removed, or otherwise edited. Once you open a new application window you will also notice that the shape of the mouse cursor changes depending on its location. When the mouse cursor is within the application window it appears as a hand holding a pen. When the mouse cursor is anywhere other than the application window it appears as an arrow.

Audio Objects

Audio objects can play wave (.WAV) files, MIDI (.MID) files, and CD audio.

Note: The Audio object uses MCI to play audio. You must have the Multimedia Extensions software to use the Audio object. In order to play .WAV and .MID files your system also requires a sound card that supports MCI, such as SoundBlaster. In order to play CD, your system requires a CD-ROM drive.

The following list shows all of the Audio object properties.

Command
CommandOnCreation
DeleteProtected
Enabled
FamilyName
Filename
Information
Layer
Length
NotifyOnComplete
NotifyOnError
ObjectData
ObjectName
PageName
PlayCount
PositionCurrent
PositionEnd
PositionSeek
PositionStart
Result
ResultString
Status
TrackCount
TrackLength
TrackNumber

Button Objects

Button objects are Push Button style by default. When you right mouse click on a Button object you can choose the Button Styles... option to show the following list of available styles:

Push Button
Check Box
Radio Button
Group Box
Picture Push Button

The selection of properties varies depending on the style of button you have chosen.

Push Buttons

A Push Button is the conventional type of button that you would use for an OK or Cancel button. You can also use Push buttons as Menu options. The following list shows all of the Push Button object properties.

Area
Bottom
ClipSiblings
ControlCommand
ControlObjectName
CursorName
DeleteProtected
Enabled
FamilyName
Filename
FilenameDisabled
Focus
Font
Height
KeyboardTabStop
Label
LabelType
Layer
Left
Location
NotifyOnClickLeft
NotifyOnClickMiddle
NotifyOnClickRight
NotifyOnEnter
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
ObjectData
ObjectName
PageName
Rectangle
Right
Size
Style
Top
Visible
Width

Picture Push Buttons

Picture Push Buttons are similar to Push Buttons that display graphics. However, the graphic in a Picture Push Button can cause the button to appear in three different states. The following list shows all of the Picture Push Button object properties.

Area
Bottom
ButtonStates
ClipSiblings
ControlCommand
ControlObjectName
CursorName
DeleteProtected
Enabled
FamilyName
Filename
FilenameDisabled
Focus
Height
KeyboardTabStop
Layer
Left
Location
NotifyOnClickLeft
NotifyOnClickMiddle
NotifyOnClickRight
NotifyOnEnter
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
ObjectData
ObjectName
PageName
Rectangle
Right
Size
Style
Top
Visible
Width

Check Boxes

Users switch Check Boxes on and off by clicking on them. Check Boxes work independently of each other. For example, if you present users with three check boxes, they can switch any number of them on or off. The following list shows all of the Check Box object properties.

Area
Bottom
Checked
ClipSiblings
ColorBackground
ColorText
CursorName
DataChanged
DataFieldName
DataObjectName
DataValueChecked
DataValueUnChecked
DeleteProtected
Enabled
FamilyName
Focus
Font
Height
KeyboardTabStop
Label
Layer
Left
Location
NotifyOnClickLeft
NotifyOnClickMiddle
NofifyOnClickRight
NotifyOnEnter
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
ObjectData
ObjectName
PageName
Rectangle
Right
Size
Style
Top
Visible
Width

Radio Buttons

Like Check Boxes, Radio Buttons switch On and Off when a user clicks on them. Unlike Check boxes, Radio buttons act as a group. For example, a group of three Radio Buttons can let the user choose among Red, Blue and Green. The user cannot select more than one option. The way to identify Radio Buttons as a group is to assign them the same FamilyName in the SmartObject Editor. The following list shows all of the Radio Button object properties.

Area
Bottom
Checked
CheckedRadioButton
ClipSiblings
ColorBackground
ColorText
CursorName
DataChanged
DataFieldName
DataObjectName
DataValueChecked
DeleteProtected
Enabled
FamilyName
Focus
Font
Height
KeyboardTabStop
Label
Layer
Left
Location
NotifyOnClickLeft
NotifyOnClickMiddle
NofifyOnClickRight
NotifyOnEnter
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
ObjectData
ObjectName
PageName
Rectangle
Right
Size
Style
Top
Visible
Width

Group Box

Use a Group Box to visually organize other objects such as check boxes and push buttons. The Group Box is different from the other button styles because users don't actually select it. Rather, it is a tool for organizing other objects in a visually helpful way. The following list shows all of the Group Box object properties.

Area
Bottom
ClipSiblings
ColorBackground
ColorText
DeleteProtected
Enabled
FamilyName
Focus
Font
Height
Label
Layer
Left
Location
NotifyOnGetFocus
NotifyOnLoseFocus
ObjectData
ObjectName
PageName
Rectangle
Right
Size
Style
Top
Visible
Width

Combo Box Objects

A Combo Box is a combination of a text box and a list box (open or drop-down). You can make the text box editable or display-only. The Combo Box serves three basic purposes: First, a Combo Box always displays a list from which a user can select an item. Second, if the text box is editable, the user can type a value. Third, if your application uses a database (via a Database object) the Combo Box can display a value from a database record. The following list shows all of the Combo Box properties.

Area
Bottom
ClipSiblings
ColorBackground
ColorSpacer
ColorText
CursorName
DataFieldName
DataObjectName
DeleteProtected
Enabled
FamilyName
Focus
Font
Height
ItemList
KeyboardTabStop
Layer
Left
Location
NotifyOnEnter
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
NotifyOnSelect
NotifyOnSelectChange
ObjectData
ObjectName
PageName
Rectangle
Right
SelectedItemData
SelectedItemNumber
ShowPartialItems
Size
Sort
TextBoxData
Top
Visible
Width

Graphic Objects

A Graphic object allows you to display a graphic on the SmartObject page. The graphics can be actual size, stretched or compressed, or tiled within a particular area. Graphic objects can be also be live or static. If you make a Graphic object live it can be changed at runtime via object icons in IconAuthor. If you make it static, it behaves as if it is part of the background and cannot be changed at runtime. The selection of properties varies depending on whether the object is live or static.

The following lists show all of the Graphic object properties.

Live Graphic Properties:

Area
Bottom
Border
BorderWidth
ClipSiblings
ColorTransparent
Command
CursorName
DeleteProtected
Dragable
DragAction
DragBringToTop
Drag Cursor
DragGraphicNo
DragGraphicYes
DragMode
DragReturnOnFail
DragTargetName
DragTransparentColor
DragType
DrawStyle
DropPosition
DropType
Effect
EmbeddedType
FamilyName
FileName
Height
Layer
Left
Location
NotifyOnClickLeft
NotifyOnClickMiddle
NofifyOnClickRight
NofifyOnDoubleClick
NotifyOnDragAbort
NotifyOnDragDrop
NotifyOnDragFail
NotifyOnEnter
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
NotifyOnPressLeft
NotifyOnPressRight
ObjectData
ObjectName
PageName
Rectangle
Right
Size
Top
Visible
Width

Static Graphic Properties:

Border
DrawStyle

Effect
EmbeddedType
FileName
SelectionArea

IconAnimate Objects

IconAnimate objects let you play animation scripts previously created with IconAnimate. You can choose to play the animation on a background bitmap or in a window. Playing an IconAnimate animation on a background bitmap makes the animation part of the background. Playing an IconAnimate animation in a window allows you to control the playing of the animation as well as the size and location of the window.

IconAnimate objects use the following properties:

Area
Bottom
ClipSiblings
Command
CommandOnCreation
CursorName
DeleteProtected
FamilyName
FileName
Height
Layer
Left
Location
NotifyOnComplete
NotifyOnEnter
NotifyOnError
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
ObjectData
ObjectName
PageName
PlayCount
Rectangle
Result
ResultString
Right
Style
Top
Visible
Width

Keyboard Objects

The Keyboard object lets you specify one or more keys or key combinations that the user can press to interact with the application. The Keyboard object is never visible at runtime. The following list shows all of the Keyboard object properties.

DeleteProtected
Enabled
FamilyName
KeyboardForward
KeyboardKeyPressed
KeyboardList
NotifyOnKeyDown
NotifyOnKeyUp
ObjectData
ObjectName
PageName

List Box Objects

List Box objects serve two basic purposes. First, a list box always displays a list from which a user can select an item. Second, if your application uses a database (via a Database object) the List Box can display a value from a database record.

The following list shows all of the List Box object properties.

Area
Bottom
ClipSiblings
ColorBackground
ColorText
CursorName
DeleteProtected
Enabled
FamilyName
Focus
Font
Height
ItemList
KeyboardTabStop
Layer
Left
Location
NotifyOnEnter
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
NotifyOnSelect
NotifyOnSelectChange
ObjectData
ObjectName
PageName
Rectangle
Right
SelectedItemData
SelectedItemNumber
ShowPartialItems
Size
Sort
Top
Visible
Width

Menu Objects

Menu objects allow you to include Windows-style menus in your applications. Although the Menu object always appears as a small icon within the SmartObject Editor, it displays as a menu at runtime. Menu objects can be top-level style or floating pop-up style. Click the right mouse button on a Menu object and choose Menu Styles... from the pop-up menu to display the choices for the different kinds of menus.

Menu object properties:

Accelerator
Caption
Checked
Enabled
FamilyName
MenuItemCount
MenuItemList
ObjectData
ObjectName
PageName
Style
Visible

Floating Pop-Up Style Menu objects also use:

Alignment
AutoTrack
MouseButton

Menu Item Object Properties:

Caption
Checked
Enabled
FamilyName
ObjectData
ObjectName
Visible

Movie Objects

The Movie object lets you play digital video and third-party animation files. For example, you can play:

Video for Windows digital video files
QuickTime for Windows digital video files
Autodesk animations
Gold Disk animations

Note: The Movie object uses MCI to play video and animation. You must have the Windows Multimedia Extensions software (which comes with Windows 3.1) to use the Movie object. You must have the proper drivers that enable the digital video and third-party animation files to run in MCI. If you are playing video or animation that uses sound, your system also requires a sound card that supports MCI, such as SoundBlaster.

The following list shows all of the Movie object properties.

Area
Bottom
ClipSiblings
Command
CommandOnCreation
ControlBar
CursorName
DeleteProtected
FamilyName
FileName
Height
Information
Layer
Left
Length
Location
NotifyOnComplete
NotifyOnEnter
NotifyOnError
NotifyOnGetFocus
NotifyOnLeave
NotifyOnLoseFocus
ObjectData
ObjectName
PageName
PlayCount
PositionCurrent
PositionEnd
PositionSeek
PositionStart
Rectangle
ResizeToFile
Result
ResultString
ReturnToStart
Right
size
status
Top
Visible
Width

OLE Objects

This section describes the OLE class of objects. OLE objects take advantage of the Microsoft Windows OLE (Object Linking and Embedding) feature. From within the SmartObject Editor you can access any Microsoft Windows application that is OLE-ready (called a server) and create data that eventually becomes an object within your IconAuthor application. The following lista shows all of the OLE object properties.

Live OLE Properties:

Area
Bottom
ColorBackground
CursorName
DefaultAction
DeleteProtected
DrawStyle
FamilyName
Height
Left
Location
NotifyOnComplete
NotifyOnEnter
NotifyOnLeave
NotifyOnStart
ObjectData
ObjectName
PageName
Rectangle
Right
Size
State
Top
TransparentBackground
Visible
Width

Static OLE Properties:

ColorBackground
DrawStyle
SelectionArea
TransparentBackground

Timer Objects

Timers allow you to use time to control your application. For example, a Timer can count up from 0 to measure how long it takes a user to perform a task. Or, a Timer can count down from a specified number of seconds to limit the amount of time a user has to perform a task. The following list shows all of the Timer object properties.

DeleteProtected
Enabled
FamilyName
ObjectData
ObjectName
PageName
Style
TimerData

System Object

The System object provides your application with key information about the kind of system the application is running on. This information is extremely valuable because the application can be running on systems with different capabilities (such as varying screen resolutions) or on systems running different operating/windowing systems.

Use the System object to find out specific information about the system the application is running on and then branch your application accordingly. For example, if your application runs on Windows and Macintosh, you can create one master application and two subapplications, one for Windows and one for Macintosh. At the beginning of the master application, an ObjGet can query the System object to find out which system the end-user is using and call the appropriate platform specific subapplication.

Besides the operating system, the System object can also find out other information about the end-users system. The following table shows the information you can get and the appropriate property to use to get it:

Information you want:	**Property to use:**
Number of Colors Available	ScreenColors
Capability to play audio and video	MultiMediaDevices
Cursor Position	CursorPostionScreen
Resolution	ScreenHeight, ScreenWidth
Which audio and video files are compatible with the system	CanPlay-

System objects use the following properties:

CanPlayCD
CanPlayMIDI
CanPlayMovie
CanPlaySound
CanPlayVideo
CursorPositionProgram
CursorPositionScreen
HasMouse
MultiMediaDevices
MultiMediaDeviceNames
PageName
ProgramType
ProgramVersion
NotifyOnLeave
NotifyOnStart
ObjectData
ObjectName
PageName
ScreenColors
ScreenHeight
ScreenPaletteEntries
ScreenWidth

Transparent Objects

The Transparent object is not visible at runtime. The object allows you to set up an area of the screen to respond to different kinds of user interaction, without changing the appearance of the information that is displayed beneath it. Typically for example, the Transparent object overlays other visual information such as a static Text, static OLE, or static Graphic object. Two common uses for the Transparent object are as a transparent Push Button and as a drop target for a dragable live Graphic object. The following list shows all of the Transparent object properties.

Area
Bottom
CursorName
DeleteProtected
DropPosition
DropType
Enabled
FamilyName
Height
Left
Location
NotifyOnClickLeft
NotifyOnClickMiddle
NofifyOnClickRight
NofifyOnDoubleClick
NotifyOnEnter
NotifyOnLeave
NotifyOnPressLeft
NotifyOnPressRight
ObjectData
ObjectName
PageName
Rectangle
Right
Size
Top
Visible
Width

Variable Objects

Variable objects let you define one or more variables and their values. As soon as the SmartObject file is loaded into memory, the variables defined in the Variable object are loaded and available for use. Although the Variable object has no visible appearance at runtime, the SmartObject file can optionally contain other objects that do appear, such as Text or Graphic objects. As an example, a Variable object might load the following:

@NUMBER_OF_GRAPHICS
3
@COLOR
blue
@CORRECT_MESSAGE
Yes. That is correct.
@INCORRECT_MESSAGE
No. That is incorrect. Try again.

As soon as the SmartObject file is loaded into memory, the variables defined in the Variable object are also loaded and available for use. This method is efficient in two ways. First, it lets you use one Display icon instead of four Variable icons. Second, it lets you load variables that are associated with a particular SmartObject display.

Variable Objects use the following properties:

Command
CommandOnCreation
DeleteProtected
FamilyName
FileName
ObjectData
ObjectName
PageName
VariableName

Text Objects

Text objects allow you to display text on the page. The objects can be display-only or interactive. The following lists show all of the Text object properties.

Live Text Object Properties:

AlignHorizontal
Area
BaseLine
Bottom
CharacterCurrency
CharacterDecimal
CharacterFalse
CharacterThousands
CharacterTrue
ClipSiblings
ColorFill
ColorFrame
ColorHighlight
ColorShadow
ColorText
CursorName
CursorNameHotword
DataChanged
DataFieldName
DataObjectName
DecimalPlaces
DeleteProtected
Editable
Enabled
FamilyName
FileName
Focus
Font
FttPageName
Height
Hotword
HotwordActivate
HotwordColor
HotwordHighlight
HotwordIndex
InputLimit
InputLimitBeep
InputTerminationRequired
KeyboardTabStop
Left
LightSource
LineSpace
Location
Mask
MultipleLines
NotifyOnClickHotword
NotifyOnClickLeft
NotifyOnClickMiddle
NofifyOnClickRight
NotifyOnComplete
NotifyOnDoubleClick
NotifyOnEnter
NotifyOnEnterHotword
NotifyOnGetFocus
NotifyOnInput
NotifyOnInputLimit
NotifyOnLeave
NotifyOnLeaveHotword
NotifyOnLoseFocus
NotifyOnPressLeft
NotifyOnPressRight
ObjectData
ObjectName

PageName
Rectangle
Right
ScrollBarVertical
Size
Text
TextCase
TextDragable
TextFormatted
TextLength
Top
Visible
Width
WidthEdge
WidthFrame

Static Text Object Properties:

FileName
SelectionArea

The Window Object

The window in which your IconAuthor application runs is actually an object, similar to the objects that are available in the SmartObject Editor.

Like the SmartObjects, the Window object has properties that your application can manipulate at runtime via object icons. The way in which the Window object is different is that you do not have to explicitly create it and you cannot delete it. As an example, the Window object has a property called TitleBarText that controls the text that appears in the Window's title bar. Although you can use a Window icon to initially set the title of the window, if you want your application to change the title at some later point, you use an ObjSet icon to set the new value of the TitleBarText property.

Note: Your application can only contain one Window object. This is different from the other objects available in the SmartObject Editor. For example, while your application can contain multiple button and List Box objects, it can only contain one Window object.

Related Topics:

Changing Window Properties
Window Appearance
Window Interactivity
Window Object Properties

Changing the Window Properties

There are several ways in which you can manipulate the Window object at runtime. First, be aware that you can successfully create an application without manipulating the properties of the Window object. However, knowing about the Window properties can help you create a more sophisticated application. For example, you can change the color of the window, change the cursor appearance, or change the appearance of the icon when the application is minimized.

The only way to retrieve or set a Window property is to use the ObjGet and ObjSet icons at runtime. Note that the Window doesn't have to exist in order to set a property. For example, the first icon in your application (even before a Window has been created) can be an ObjSet icon that sets the cursor to an I-beam. As soon as an icon executes that displays information on the screen the Window is created and the new cursor property is in evidence.

Window Appearance

The CursorName property controls how the cursor appears over the window background. By default, the cursor is set to appear as an arrow. Set the CursorName property to make the cursor appear. The CursorName property can be set independently for the Window object and for SmartObjects, allowing you to vary how the cursor appears depending on the item it is positioned over.

The ColorBackground property lets you set the color of the window. Re-setting this property has the same effect as using a Clear icon to paint the display. The TitleBarText property lets you set the text that appears in the window's title bar. Control the way the window looks when it is iconized via the IconFileName property.

Use the Maximized and Minimized properties to control the state of the window. For example, by setting Minimized to True, you can cause the window to be iconized.

Set the ClipChildren property to True if live objects appear to flash when information is displayed on the Window object. In order to set ClipChildren to True, you must include an ObjSet icon and a Window icon in your structure. The ObjSet icon sets the property and the Window icon creates the window.

Each live object you display via a SmartObject page is, by Windows definition, a window in itself. Specifically, these objects are called child windows because they are the "children" of the Window object on which they are displayed. Whenever possible, IconAuthor automatically clips child windows. When a child window is clipped, it means that the area is unaffected by information being displayed on the rest of the background. As a result, for example, if a Button object displays and then a Display icon fades a graphic onto the Window background, the Button is unaffected by the fade because it is clipped. If the Button was not clipped, it would flash repeatedly as it attempted to re-draw itself in the wake of the fade-in graphic display.

In most situations IconAuthor recognizes when it has to clip child windows. There are however, some situations where child windows are not automatically clipped. In these cases, you need to explicitly set the ClipChildren property to True. (The property is False by default.)

IconAuthor does not automatically recognize and clip child windows when third party program information is displayed in the window. For example, if your application displays a Button object and then displays a Gold Disk animation file on the background, the Button flashes. The Button flashes because it is not automatically clipped and it is re-drawing itself to remain in view on top of the animation. In this same scenario, if you set ClipChildren to True, the Button does not flash because it is clipped.

Window Interactivity

Like a button, the Window object has NotifyOnClickLeft, -Middle, and -Right properties that let your application detect when a user clicks on the window background. The object also has the NotifyOnPressLeft property which lets your application detect when the user presses the left mouse button on the background. If you are designing your application for a touch screen, this same property also generates an event when the user touches the window background.

Properties such as NotifyOnMinimize and NotifyOnSize generate an event when the state of the window changes. For example, if NotifyOnMinimize is True and the user minimizes the window, a "Minimize" event occurs.

Window Object Properties

The Window object uses the following properties.

Area
Bottom
ClipChildren
ColorBackground
CursorName
Enabled
FitToWindow
Focus
Height
IconFileName
Left
Location
Maximized
Minimized
NotifyOnClickLeft
NotifyOnClickMiddle
NofifyOnClickRight
NotifyOnMaximize
NotifyOnMinimize
NotifyOnPressLeft
NotifyOnSize
ObjectName
Rectangle
Right
Size
TitleBarText
Top
Width

Content Editors

Every icon has a dialog box associated with it called a **Content Editor**. Adding content to an icon is the process of entering the information requested by its Content Editor. You can add content to the Content Editor of an icon in the structure or in the library. If you add content to an icon in the structure you change only that instance of the icon. If you add content to an icon in the library each subsequent time you build a copy of it into the structure, that copy has the edited characteristics.

Note: When you add content to an icon in the library it does not affect any instances of that icon that you have previously built into the structure.

Adding content is a flexible part of the authoring process. You can add content to an icon before you build it into the structure (while it is still in the library). You can add content to an icon immediately after you build it into the structure. Or, you can build several icons into the structure and add content to them later.

Related Topics:

Opening Content Editors
Closing Content Editors
Naming Icons
Entering Values
Accessing Editors or Dialog Boxes
Types of Content Editor Values

Opening Content Editors

There are three general techniques for opening Content Editors.

Using the Mouse

To open a Content Editor with the mouse:

+ Double-click on the icon.

Using Menu Commands

To open a Content Editor with menu commands:

1. Click on the icon (in the library or the structure) to select it.
2. Pull down the Edit menu.
3. Choose Library or Application.
4. Choose Add Content...

Opening Automatically upon Building

To open a Content Editor automatically when an icon is built into the structure:

+ Choose Add Content On Build from the Options menu.

  The feature is toggled on. Until you choose the Add Content On Build command again to toggle it off, every time you build an icon into a structure, its Content Editor is automatically opened.

Note: You can only open one Content Editor at a time. If a Content Editor is open you cannot perform any other activities in IconAuthor until you close it.

When you open an icon, its Content Editor is displayed. The dialog box contains a number of text boxes that you fill in to define the icon content. Some of the text boxes contain default values which will be used if you do not specify alternatives.

Closing Content Editors

When you finish adding content to an icon, you close its Content Editor.

Closing and Accepting Changes

To close a Content Editor and accept the values:

+ Choose OK.

Your changes will be saved and the icon will be changed to grey when it is not selected unless it is the lead icon in the composite.

Closing and Cancelling Changes

If you decide not to enter values or not to accept values that you have entered into text boxes, you can close the Content Editor without accepting any changes.

To close a Content Editor without accepting changes:

+ Choose Cancel.

Closing a Composite

If the icon you are adding content to is the lead icon in a composite, when you choose Cancel, the Content Editor of the next icon in the composite is automatically opened. To close that Content Editor, choose Cancel again. Each time you choose Cancel, until you reach the last icon in the composite, the next Content Editor in the composite is opened.

To stop adding content to all composites in a range:

+ Choose Exit Range.

Icon Colors

If IconAuthor runs in black and white on your display, an icon is highlighted when you select it. If you are running IconAuthor in color, icons appear in different colors. The color of an icon varies depending upon whether it is selected, whether it has content added to it, and whether it is a single icon or a composite. Use the icon colors to help you keep track of the icons to which you have added content.

This icon color:	**Means:**
blue	The icon is selected
gray	The icon has content and is not selected
yellow	The icon does not have content and is not selected.
green	The icon is the lead icon in a composite and is not selected.
black	The icon is disabled and is not selected.

Naming Icons

All single icon Content Editors have an Icon Name text box. The value in the Icon Name text box is the label that appears below the icon in the structure. Initially, the value is the generic name of the icon.

You can type a new icon name that is more meaningful to you. For example, you might include a Pause icon in your structure that causes the application to pause for 2 seconds before executing the next icon. Initially, the icon is named "Pause". You can use the Icon Name text box to rename the icon "2 Sec". At a glance, when you see the icon in the structure you will be reminded of its function.

Similarly, composite icons contain a Composite Name text box that allows you to name a composite anything you'd like. As an example, if your composite is several icons that together allow a user to log into your application, you can rename the composite "Login."

Entering Values

There are three ways to enter values into Content Editor fields. You can type them, or frequently, you can use drop-down list boxes.

Typing Values

If you know the value you want to enter in an empty text box you can simply click in the text box and type the value. If the text box already contains a value, select the existing value by double-clicking on it (if its not already selected) and begin typing the new value. As you begin typing, the old value is automatically removed.

Using Drop-Down List Boxes

Most Content Editor text boxes have corresponding drop-down list boxes from which you can select an item.

To use a drop-down list box:

1. Click on the small down arrow detached and to the right of the text box.

   The drop-down list is displayed.

   The default value, if there is one, is highlighted. If there are more items than can fit in the list box, the list box has a scroll bar.

2. Click on the item you want to select.

   The drop-down list box closes and the selected item appears in the text box. The result produced when you select an item depends on whether the item is a *value* or an *editor* or *dialog box*.

Accessing Editors or Dialog Boxes

Some Content Editor list box items access one of several tools (either an editor or specialized dialog box) to help you locate or create a value for the text box. An example of a drop-down list box item is the SmartObject Editor. When you choose this item the SmartObject Editor appears, allowing you to create or edit a file to display in your application.

Special editors and dialog boxes are available through certain icons where appropriate. For example, one of the functions of the Display icon is to display a SmartObject file, therefore the Display icons Content Editor lets you access the SmartObject Editor. Whenever an item accesses an editor or special dialog box, the item name is followed by ... (as in for example, SmartObject Editor...).

Use an editor or dialog box to create or edit values for the Content Editor field. When you close the editor or dialog box, the value that you chose is automatically returned to the Content Editor text box. Several editors, like the Animation Editor, return a filename to a Content Editor text box. Other editors, including the Color Editor, return other kinds of values to a Content Editor text box, such as the number that represents the selected color.

Values for Content Editor Fields

The following kinds of values are typically used by Content Editor fields:

Numbers
Strings
Coordinates
Filenames
Keywords
Colors
Variables

Numbers

Several Content Editor fields require a numeric value. Consider the Pause icon which pauses execution for a specified period of time. It has a field called Number of Seconds in which you enter a number to denote how seconds the pause should last. Fields that require numbers typically have a default value and a selection of frequently used values in the drop-down list. You also have the option of typing a number of your own choosing.

Strings

Strings are values such as names, address, descriptions, words and sentences that are made up of alphanumeric characters. Here are some examples: green, George Watkins, and 31 Maple Street. Although strings can contain numbers, they are not intended for calculations. As an example, consider the MsgBox icon which is designed to display a message on the screen. It has a field called Message in which you specify a string that you want to appear.

Coordinates

Several Content Editor fields require you to specify a position on the screen where you want something to occur or display. Consider the Display icon which is used for displaying graphic, animiation or SmartObject files. The Display icon has a field called Location that you use to indicate exactly where you want the file to appear.

To specify a position on the screen, you need to understand that your screen is actually made up of tiny dots called pixels. When you see an image on the screen, the effect is the result of different patterns and colors of pixels. Different screens are made up of different numbers of pixels. The more pixels, the sharper the display of an image can be. This degree of sharpness is called the **resolution** of the screen. Resolution is expressed as the number of horizontal dots (in a single row) by the number of rows. For example, if each row of your screen is made up of 640 pixels and it has 480 rows, your screen resolution is 640 x 480.

When you need to specify a location on the screen, you describe that point by giving its X and Y coordinates. The X coordinate indicates how far to the right the point is and the Y coordinate indicates how far down the point is. The top, left corner of the screen is 0,0.

Going back to the Display icon example, if you specify a Location of 0,0 the file will appear in the top left corner of the screen. If you specify 100,100 the file will appear exactly 100 pixels down (from the top) and 100 pixels over (from the left side) of the screen.

Filenames

Some Content Editors require a filename. For example, the Display icon requires the name of the file to display and the Program icon requires the name of the program file to execute. You can type a filename or use the Directory item from the drop-down list to find a filename. The Directory item is a special dialog box that lets you search through drives and directories to find the correct file. When you find the file you want to use, you double-click on it and the filename is automatically returned to the appropriate Content Editor field.

Keywords

Many Content Editor fields will only accept a specific variety of values called keywords. For example, the Beep icon (which generates a beep sound) lets you choose between the values High or Low depending on the type of sound you want to use. In these situations, you can type a value or select one from the drop-down list.

Colors

When a Content Editor field requires a color, it accepts the value in two basic forms. You can specify a keyword, such as green, white or yellow or you can specify an **RGB** value. An RGB value is made up of three numbers separated by commas, such as 0,255,127. Each number represents the level of intensity of red, green, and blue, respectively, that make up the resulting color. RGB values allow you to use a large number of colors that are subtle variations on the limited colors represented by keywords. For example, while the yellow that is represented by the keyword yellow has an RGB value of 255,255,0, you could specify a shade of yellow that has slightly more blue in it with the value 255,255,10.

Content Editor fields that require a color value let you access a Color Editor. Because the editor lets you choose from all possible color variations, it automatically returns an RGB value to the Content Editor field.

Editing Structures

When you edit the structure of your application, you select icons within the structure, remove them, or copy them, and insert them elsewhere in the structure, or in a different structure. By dragging and dropping icons you can cut or copy them to the Clipboard. The Clipboard is an off-screen holding area represented by the Clipboard on the ribbon bar. The appearance of the Clipboard varies depending on whether it contains an icon.

If the Clipboard contains one or more icons you can drag a copy of the Clipboard from the ribbon bar to paste the previously cut or copied icon(s) elsewhere in the structure. The Clipboard can only hold one icon or one group of icons at a time. When you cut or copy an icon (or group of icons) to the Clipboard it replaces the icon (or group) that was there previously.

You can cut, copy, and paste all within one application window, or you can open another application window and use these editing functions between different windows. For example, you can open an existing application and a new application in the work area at the same time. If you are satisfied with a part of the existing structure and want to use it in the new structure, you can simply copy those icons and not recreate them.

Related Topics:

Selecting Icons
Cutting Icons
Copying Icons
Pasting Icons
Clearing Icons
Inserting Another Application into the Current Structure
Copying a Part of a Structure to a New File

Selecting Icons

Selecting a Single Icon

To select an icon:

+ Click on the icon.

If IconAuthor is running in color a selected icon is blue. If it is running in black and white the icon is highlighted.

Selecting a Range of Icons

To edit a range of icons you first select the range. After the range is selected you can perform editing tasks such as cutting an icon, just as if you were working with only one icon.

To select a range of icons:

1. Click on the first icon in the range you want to select.
2. Hold down SHIFT while you click on the icon you want to be last in the selected range.

Note: To select a range of icons that makes up a composite, click on the lead icon. The other icons in the composite are automatically selected.

If IconAuthor is running in black and white a selected range of icons is highlighted. If it is running in color a selected range is blue.

Cutting Icons

When you cut icons you remove them from the structure and place a copy on the Clipboard.

To cut icons from the structure:

1. Position the mouse pointer over an icon or a range of selected icons.

   The cursor is shaped like a hand holding a pen.

2. Press and hold the mouse button.
3. Drag to the Clipboard.

   As you drag toward the Clipboard the cursor often appears as a red circle with a diagonal bar through it. Whenever the cursor appears in this form it means that the current position is not a valid drop location.

4. Position the cursor over the Clipboard.

   If you are dragging a single icon the cursor appears in the form of the cut icon. If you are dragging a range of icons the cursor appears as cascading icons.

5. Release the mouse button.

   The icon or icons are removed from the structure and copied to the Clipboard. A small square appears on the Clipboard to indicate that the Clipboard has an item in it.

Note: You can also use a menu command to cut icons. Select the icon or range of icons and choose Cut from the Edit menu.

Copying Icons

When you copy an icon (or a group of icons) it is unchanged in the structure and a copy of it is placed on the clipboard.

To copy icons from the structure:

1. Position the mouse pointer over the icon or over a range of selected icons.
2. Press and hold the CTRL key and press and hold the mouse button.
3. Drag to the Clipboard.
4. Position the cursor over the Clipboard.

   If you are copying a single icon the cursor appears in the form of the copied icon. If you are copying a range of icons the cursor appears as cascading icons.
5. Release the mouse button.

   The icon or icons are copied to the Clipboard. A small square appears on the Clipboard to indicate that the Clipboard has an item in it.

Note: You can also use a menu command to copy icons. Select the icon or range of icons and choose Copy from the Edit menu. A copy of the icon or range of icons is automatically placed on the Clipboard.

Pasting Icons

When you paste an icon (or a group of icons) you insert a copy of the icon(s) stored in the Clipboard, into the structure. You can only paste an item if it was previously cut or copied. A small square (that represents an icon) appears on the Clipboard to indicate that the Clipboard has an item in it.

To paste one or more icons:

1. Press and hold the mouse cursor on the Clipboard.

   The cursor appears in the shape of the Clipboard.

2. Drag to the position in the structure where you want to paste.

   When you drag to a valid drop position the cursor again appears as the Clipboard.

3. Release the mouse button.

   A copy of the icon(s) currently stored in the Clipboard is inserted into the structure.

Note: You can also use a menu command to paste icons. Choose Paste from the Edit menu. The cursor appears in the form of the Clipboard. Move the cursor to a valid drop position in the structure and click.

Typically, you paste below an existing icon in the structure. However, in some cases, as with the If icon, the Module icon, or any lead icon in a composite, you can also paste to the right.

Clearing Icons

When you clear an icon you permanently remove it from the structure by dragging it to the trash can on the ribbon bar. A copy of the icon *is not* placed on the Clipboard. Use this function with care because icons that are cleared from the structure are not recoverable.

To clear icons from the structure:

1. Position the mouse pointer over the icon or over a range of selected icons.
2. Press and hold the mouse button.
3. Drag to the trash can.

   The cursor appears in the form of the icon.
4. Release the mouse button.

   A dialog box appears that asks you to confirm the clear action.
5. Choose Yes to continue or choose No to cancel the action.

   When you choose Yes the icon is removed from the structure.

Note: You can also use a menu command to clear icons. Select the icon or range of icons and choose Clear from the Edit menu. The selected icons are automatically removed from the structure.

The dialog box that asks you to confirm the clear is optional. When IconAuthor is first installed, the feature is selected and a check mark precedes it on the menu. If you do not want the dialog box to appear choose Confirm Clear from the Options menu. The check mark will be removed and the feature is de-selected.

Inserting Another Application into the Current Structure

IconAuthor lets you insert all of the icons from another application into the current application.

To insert one application structure into another:

1. Choose Paste From... from the Edit menu.

   The Paste From dialog box appears.

2. Use the dialog box to select the application file you want to insert.

3. Choose OK.

   The Paste From dialog box is closed. The active window reappears. The cursor appears in the shape of the Clipboard.

4. Move the cursor to the location where you want to insert the icons.

5. Click to insert the icons.

Note: The structure you pasted was also copied to the Clipboard. It remains there until you copy something else over it. By dragging the Clipboard to a position in the structure and releasing, you can re-insert the same structure of icons repeatedly.

Copying Part of the Current Structure to a New File

You can copy (and save) part of the current application to a new file.

To copy icons and save them to a new file:

1. Select one icon, a range of icons, or all the icons in a structure.

   To select all the icons, choose Select All from the Edit menu.

2. Choose Copy To... from the Edit menu.

   The Copy To dialog box appears.

3. Enter the name of the file you want to create (and to which you want the icons copied).

4. Choose OK.

   The new file is created (not opened).

Note: If you select a file name that already exists, the range of icons copied will replace the previous contents of the file.

Running an Application

To determine whether your application is performing as you intend, run it. You can run all or part of the application, including an individual icon. You can also disable icons in the structure and run the application without them.

The IconAuthor window is hidden when an application runs. When an application finishes running the IconAuthor window comes back into view. If your application has an error in logic that causes it to loop endlessly, press ESC to return to the IconAuthor window.

Related Topics:

Run Techniques
Hints for Running from a Selected Icon
General Execution Rules

Run Techniques

Use the run buttons on the ribbon bar to run the application. The left run button is grey and runs the application from the top. The right run button is light blue if you are using the IconAuthor default color scheme and runs the application from a selected icon.

Note: It is strongly recommended that you save your application before you run it

Running the Entire Application

To run an entire application:

+ Click on the left run button in the ribbon bar.

Running from a Selected Icon

To run the application from a selected icon:

1. Click on an icon to select it.

   Note that if you click on a lead icon in a composite, the entire composite is selected. Even so, the application will run from the lead icon (the first selected icon).

2. Click on the right run button in the ribbon bar.

Note: You can also use the Run menu commands to run the application. Choose Run From Top or select an icon and choose Run From Selected.

Halting a Running Application

To stop an application from running:

+ Press ESC.

Hints for Running from a Selected Icon

If you run your application from a selected icon, you may find that certain icons (above the selected icon in the structure) that are critical to the application are not executed. For example, consider the Window icon which lets you define what kind of window the application will run in. Many applications use a Window icon as one of the first in the structure to specify whether the application should run full screen or in a smaller window. If you run from a selected icon and the Window icon (or another critical icon) is not executed, your application may not run as expected.

Related Topics:

The Startup Icon
The Default Window Setup

The Startup Icon

If you include a Startup icon as the first icon in your application you can use it to identify key icons that you always want to run, even if you are running from a selected icon further down in the structure. Regardless of the position of the selected icon, the icons marked by the Startup icon are executed first.

To use a Startup icon:

1. Build the Startup icon into the top of the structure.
2. Build an icon that you always want to execute first, to the right of the Startup icon.
3. Build any other icons that you want to execute first, below the icon in step 2.

The figure to the left shows an example of a Startup icon that directs an application to execute a Window icon and a Help icon (to make online Help available) even if you run from a selected icon further down in the structure.

The Default Window Setup

The Default Window Setup option is another way to avoid missing a critical Window icon when you run from a selected icon further down in the structure. It lets you specify, for example, whether the application should run full screen or in a particular size window.

Set up the Default Windows Setup to match the performance of the Window icon at the beginning of your structure. This way, your application will run properly even when the Window icon does not execute.

Important: The Default Window Setup only controls how the application runs when you are editing in IconAuthor. If you want the application to run in a window (versus full screen) when you distribute it to end-users, you must use a Window icon.

To change the defaults for how an application appears when it runs:

1. Choose Default Window Setup... from the Run menu.

   The Default Window Setup dialog box appears.

2. Use the Size text box to change the size of the window.

   Use the drop-down list to select Full Screen or access the Area Editor to define the area of a window you prefer to use.

3. Use the Title text box to assign a title to the window.

   If you specified Full Screen as the size of the window, a value in the Title text box is disregarded. (Full screen applications do not have title bars.)

4. Turn on the Re-size option if you want the window to be resizable.

   If you specified Full Screen as the size of the window, the Re-size option is disregarded.

5. Change the background color of the screen.

   The drop-down list box provides a selection of colors and also lets you access the Color Editor to create or choose an alternative color.

6. Choose OK to accept changes to the dialog box, or choose Cancel to close the dialog box without accepting changes.

General Execution Rules

Execution flow is the order in which icons are executed when you run an application. There are several general rules that you should be familiar with in order to understand why your application executes as it does. Many of these rules are related to the way specific icons behave, particularly the icons that are in the Icon Librarys Flow folder.

Rules:

Simple Downward Execution Flow

If there is an icon below the current icon, and there is no icon to the right, execution flows downward to the next icon.

Composite Execution Flow

A composite is a single icon in the library that typically creates a mini-structure of icons when you build it into a structure. Composites are convenient because they consist of icons that are commonly used together. The first icon in a composite (the one that appears in the library) is always green and is called the lead icon. When IconAuthor encounters a lead icon it always executes the icon to the right next.

As an example, in the figure to the left, the LoopIndex icon is the lead icon in the LoopIndex composite. First, execution flows downward from the Start icon to the LoopIndex icon. Second, instead of immediately continuing downward to the Display icon, execution flow automatically takes a right turn to the LoopStart icon. From the LoopStart icon, execution once again flows downward to the Display icon and the LoopEnd icon. When all the icons in a composite have finished executing, execution flows to the icon below the lead icon in the composite. In the example to the left, after the icons in the composite complete, the Display icon executes.

Application Completion

You can build a deliberate exit into your application using an Exit icon. However, if IconAuthor runs out of icons to execute, the application will automatically end. For example, in the preceding figure (used to demonstrate Composite Execution Flow) if there are no icons below the Display icon (below the LoopIndex icon), the application ends.

Calling other IconAuthor Applications

Your application is sometimes referred to as a **main application** because it can use several icons (for example, SubApp, Subroutine and Help) to run other IconAuthor applications. When IconAuthor executes one of these icons, it interrupts the typical flow of execution by jumping to an entirely different file (and structure). If the called application returns control to the main application, execution flow continues downward from the icon that was last executed. In the example on the left, a SubApp icon runs a another application (called a sub-application). When the sub-application completes, execution flow returns to the main application.

Note that this is a simple example. As shown in the following figure, the user could exit the application entirely via the sub-application (if you give them the opportunity). In this situation execution flow never returns to the main application.

Also, in an even more complex situation, one sub-application may call another. If your application uses this kind of logic you must be careful to keep track of the various sub-applications. As execution flows from one sub-application to another, these files are loaded into memory. If the application flows to but never returns from too many (over 99) sub-applications, an error occurs.

Exceptions to Basic Execution Flow

There are several icons that represent some of the most common exceptions to the typical rules of execution flow.

Exit Icon
Loop and LoopIndex Composites
If Icons

Exit icon

When IconAuthor encounters an Exit icon, execution does not flow in a simple downward direction. This icon lets execution flow back out of a part of a structure. For example, depending on how you set up its content it can let execution flow from a sub-menu back to a main menu, from a sub-application back to a main application or from a main application out to the windowing environment.

Loop and LoopIndex Composites

You can create a loop in your application using either the Loop composite or the LoopIndex composite. A loop is part of your structure that executes repeatedly. All the icons that you build within a loop execute repeatedly. The figure on the left shows how a Loop composite appears when you first build it into a structure.

The figure on the right shows how the same composite looks when it is set up to execute a Clear and Display icon repeatedly.

Use the Loop when you want *the user* to control how many times the structure repeats; use the LoopIndex when *you* want to control how many times the structure repeats.

Like the Exit icon, the Loop and LoopIndex composites vary from the typical execution flow. Execution flows normally except for the LoopEnd icon. When IconAuthor encounters a lead Loop or LoopIndex icon execution flows immediately to the right. From the LoopStart icon, execution then flows downward. When the LoopEnd icon is executed, instead of going back up to the lead icon and dropping down, execution flows back to the LoopStart icon.

If Icon

An If icon gives your application the opportunity to branch, that is, take one path versus another. Each If icon is set up to test whether a specific condition is true. (For example, it can contain content that, in effect, asks the question, Is the user over 20 years of age?) When IconAuthor encounters an If icon it evaluates whether the stated condition is true. From the If icon, execution either flows downward if the condition evaluates to True or to the right if the condition evaluates to False. This is illustrated in the figure on the left.

Debugging

When you debug an application, you detect and correct errors in its structure and/or content. IconAuthor lets you debug all or part of your application as you are developing it.

The following tools are useful for debugging in IconAuthor:

Temporarily excluding (disabling) icons from execution
Understanding error messages
Viewing the contents of variables
Hints for debugging large structures

Important: You can also use the IAScope viewer to debug an application. For more information see the IAScope program.

Disabling and Enabling Icons

IconAuthor lets you disable and enable icons to help you debug your application. If your application contains a bug and you are not sure of the source of the problem, disabling icons can be helpful. As an example, you can disable all but one icon or a small group of icons. Run the application to be certain this limited part of the structure is performing correctly. If it is, enable more icons and eventually determine the source of the bug. Use the disable and enable buttons in the ribbon bar.

To disable icons:

1. Select one icon or a range of icons.
2. Click on the disable selected icons button.

To enable selected icons:

1. Select one disabled icon or a range of disabled icons.
2. Click on the enable selected icons button.

To enable all icons in a structure:

+ Click on the enable all icons button.

Using Error Messages

In some situations an error in structure or content may cause IconAuthor to display an error message. If this occurs, refer to the description of error messages in the IconAuthor Reference Manual.

Viewing and Editing Variables

Variables, which can lend great flexibility to your applications, can also serve as a powerful tool in debugging. After running an application, simply viewing the contents assigned to a variable provides an indication of what's happening "behind the scenes." In addition to viewing the structure of an application, IconAuthor lets you run an application and then view all of the variables (and their values) used by the application.

As an example of how this might be useful, consider an application that is supposed to let the user input his or her name, store it in a variable called @NAME, and then redisplay the name (the contents of @NAME) later in the application. Perhaps you find when you run the application that the users name does not appear as expected. In this case, you could begin tracking down the problem by viewing the applications variables and seeing whether @NAME is being used at all, and if so, whether the correct value is being assigned to it.

Related Topics:

Types of Variables to View
Changing the Window View
Viewing Structures and Variables Simultaneously
Editing Variables
Clearing Variables

Types of Variables

IconAuthor lets you view three different kinds of variables: user, system and path. User variables are those that you specifically create for your application, like @NAME or @COLOR. System and path variables are reserved for special functions and are assigned values by IconAuthor during execution. For example, a path variable called @_GRAPHIC_PATH is always set to the path and directory where IconAuthor can find the graphics for the current application. @_GRAPHIC_PATH would therefore contain a value similar to C:\IAUTHOR\GRAPHICS. Like the user variables, it is often useful to be able to view the contents of system and path variables.

Changing the Window View

By default, the application window shows the structure. You can change the view so that it displays a type of variable used by the application instead.

To view the variables of an application:

1. Ensure that your application window is active by clicking anywhere on it.
2. Choose Window Contents from the View menu or click on the Window Contents button in the ribbon bar.
3. Choose the desired command from the menu.

The application window changes to display the appropriate list of variables and their current values. Note that user variables appear in the list only after you have run the application during this editing session and they contain values.

Viewing Structure and Variables Simultaneously

You may find it even more helpful to view several perspectives of one application simultaneously. IconAuthor lets you duplicate a window and then change the view presented in that window. For example, you can create a structure in an application window, duplicate the window, and use the previously described procedure to change the view of the duplicate window to show the system variables for that application. You can then tile the windows so that they appear side by side.

To duplicate a window:

1. Click on the window you want to duplicate to make it active.
2. Choose Duplicate from the Window menu.

   A duplicate of the active window is created. The duplicate window has the same name in the title bar as the original window, except that the name is preceded by "Duplicate:"
3. Use the Window Contents commands in the View menu (or the Window Contents button) to change the perspective of the duplicate window.

If at any time you want to find out exactly which document windows you have open, pull down the Window menu. The bottom of the menu lists the currently open windows. The active window is preceded with a check mark.

Editing Variables

You can change the value currently assigned to a variable by highlighting the variable, typing a new value in the box at the top of the window, and pressing the RETURN key.

Clearing Variables

It is often useful to be able to clear all of the application variables used by an application. For example, you might run your application and then view its variables to see what they contain. Typically, in fine-tuning and debugging your application you will make some changes to the structure. Before running the application again, you have the option of re-setting all the application variables to their original state (so that they do not contain values).

To clear application variables:

1. Click on the Variables button in the ribbon bar.

   A pop-up menu appears.

2. Choose Clear Application Variables.

   The variables of the currently active application are re-set to their original state.

Debugging a Large Structure

There are several features that are part of the IconAuthor interface that help you to edit a particularly large structure that cannot be viewed all at once.

Related Topics:

Zooming the Structure
Compressing Composites
Jumping to Icons in a Large Structure

Zooming the Structure

When you first run IconAuthor the size of the icons is considered 100%. You can zoom the icons in the structure (not in the library) so that you view them smaller at 75%, 50%, or 25%.

To change the zoom level of icons in the structure:

1. Press and hold the mouse on the zoom button in the ribbon bar.
2. Drag to select a zoom level.

Note: You can also use the View menu to change the zoom level. Choose Zoom from the View menu. Choose a zoom level percentage.

You can add additional zoom percentages to your menu. For instructions on how to do so, refer to the section on IAUTHOR.INI in Appendix C.

Compressing Icons in a Composite

Another way to see more of the structure at once is to compress (hide) the icons in selected composites.

To compress the icons in a composite:

1. Select the lead icon in the composite.
2. Click on the compress/expand composite button in the ribbon bar.

   The composite is compressed. (Select the icon and click the compress/expand composite button again to show the icons in the composite.)

Jumping to Icons in a Large Structure

Instead of scrolling through the IconAuthor window to find a part of the structure you want to edit, you can use a search feature to automatically jump to a particular icon and select it.

To jump to an icon and select it:

1. Choose Find from the Edit menu of the IconAuthor window.

   The Find Icon dialog box appears.

2. Enter the name of the icon in the Icon Label text box and choose OK.

If the icon you specified appears in the structure (below or to the right of the currently selected icon), IconAuthor locates the icon and selects it. You can now copy or cut the icon or move on to another building or editing task. If you do not customize the names of your icons, it is likely that you will have several icons with the same name in your application. IconAuthor will find the first instance of that icon name the first time you search. To find the next one, choose Find Next from the Edit menu or press F3.

Managing Files

Use the following procedures to manage files when you work with IconAuthor:

General Procedures

Naming Files
Organizing Files
Using Page Setup
Printing
Exiting from IconAuthor

Application Files

Opening Existing Application Files
Auto Saving Application Files
Distributing Applications
Saving Applications

ASCII Text Files Files

Opening Existing ASCII Text Files
Starting to Create ASCII Text Files
Saving ASCII Text Files

Graphic Files

Opening Graphic Files in the IconAuthor Work Area

SmartObject Files

Opening SmartObject Files in the IconAuthor Work Area

Editing between Windows

Each new or existing file you open and work with is contained in its own document window. You can therefore open and work with as many files as necessary at the same time. For example, you can create an application window and begin building a structure. You can open an existing application in another application window and perform structural edits between the two windows. While the application windows are still open you can open a text window and do some work on a Variable file (an ASCII text file). At a later point, when you are considering which graphic to include in a Display icon, you can open several graphic files to view them.

Manipulate document windows (similar to the way you work with application windows) by moving them, resizing them, or minimizing them within the IconAuthor work area. When you minimize a document window the icon is different depending on the type of file the window contains. For example, the icon for a minimized application window is a hand using a pen to create a structure. The name of the file in the document appears below the icon. You can also show path names for minimized files, by choosing Show File Path from the Window menu.

Related Topics:

Editing Applications

Naming Files

IconAuthor expects authors to use the following file naming conventions:

- Main application files have an .IWM extension.
- Sub-application, Subroutine, and Help files have an .IW extension.
- ASCII Text files have a .VAR, .PTH, or .TXT extension.
- SmartObject files have an .SMT, .FTT, or .TXT extension.
- Graphic files have one of the following extensions:

.ATT	.FIF	.KFX	.PSD	.XPM
.BMP	.GIF	.LV	.RAS	.XWD
.CAL	.GX2	.MAC	.RLE	
.CLP	.ICA	.MSP	.TGA	
.CUT	.ICO	.PCD	.TIF	
.DCX	.IFF	.PCT	.WMF	
.DIB	.IMG	.PCX	.WPG	
.EPS	.JPG	.PIC	.XBM	

Related Topic:

Organizing Files

Organizing Files

IconAuthor provides a default directory structure to help you organize the files that make up your applications. Store application files in one subdirectory, graphic files in another, etc. It is important to understand this directory structure to make the most effective use of IconAuthor. For example, if you use this system to store a file such as a graphic file, IconAuthor will readily know where to find that file when it is time to include it in an application at runtime.

During installation, IconAuthor automatically creates a main directory called IAUTHOR unless you specify a different name, such as, MYWORK. For simplicity,we assume that your main IconAuthor directory is called IAUTHOR. The IAUTHOR directory contains all the files that allow IconAuthor to run.

Also at installation, IconAuthor creates subdirectories below the IAUTHOR directory. These subdirectories are used when you save and open files. For example, save application files in the ICONWARE subdirectory and graphic files in the GRAPHICS subdirectory. It's important that you put your files in these subdirectories unless you want to enter path information each time you enter a file name or change your path file to reflect the location of the files. That is where IconAuthor *expects* to find them. As an example, you might indicate in a Display icon Content Editor that at runtime IconAuthor should display EAGLE.PCX. Even though you don't specify a path for the graphic file, IconAuthor expects to find EAGLE.PCX in the GRAPHICS subdirectory.

The following list describes where to save various types of files:

ANIMATE	Iconanimate animation files
AUDIO	audio files
DATABASE	database files
FORMAT	database format files
GRAPHICS	graphic files
ICONWARE	application files
INPUT	input menu template files
LIBRARY	icon library files
MOVIE	digital video and third-party animation files
TEXT	ASCII & SmartObject files
VARIABLE	variable files

Starting to Create ASCII Text Files

The IconAuthor File menu lets you create new ASCII text files.

Note: You can also use a text editor such as Notepad to create and edit ASCII text files. To access Notepad from within IconAuthor, choose Editors from the Run menu and then choose Notepad. For information on using Notepad, refer to your Microsoft Windows documentation.

To open a new application or text window:

1. Choose New... from the File menu.

   The New dialog box appears.

   By default, the Main Application option is selected.
2. If necessary, select the Text Window option.
3. Choose OK to open a new window.

   An untitled window appears with a vertical bar in the upper left corner to indicate where text appears when you begin to type. Use the keyboard to create the contents of the text file.

Related Topic:

Editing ASCII Text Files

Editing ASCII Text Files

Your application may require that you create or edit one or more of the following types of ASCII text files:

- variable files (used by the LoadVar Icon and SaveVar Icon)
- straight text files (used by the Text Icon)
- database format files (used by the Database Icon)
- path files (used by IconAuthor to find the files used by your application)

You can edit a text file in an IconAuthor text window or in another editor such as Windows Notepad.

Using Notepad

You can access Notepad from within IconAuthor by choosing Notepad from the Run menu. For information on using Notepad, refer to your Notepad documentation.

Using an IconAuthor Text Window

When you open an IconAuthor text window, a vertical bar appears in the upper left corner to indicate where text appears when you begin to type. Use the keyboard to create the contents of the text file. Press RETURN to start a new line. (Text does not automatically wrap to the next line.)

When you want to change existing text, backspace to remove text one character at a time. You can also use the mouse to select any quantity of text and use the Cut, Copy, Paste, and Clear commands from the Edit menu. When you cut or copy text it is placed on the Clipboard. Previously cut or copied text can then be pasted into a new position in the text window.

Related Topics:

Creating ASCII Text Files
Saving ASCII Text Files

Opening Existing Application Files

When you open an existing application file, the file is placed in a new document window. You can edit the structure and content of an application file. If two application files are open at the same time, you can edit between them. For example, you can perform editing tasks such as copying an icon from one structure to another. You can also print application files.

.**To open an existing file:**

1. Choose Open... from the File menu.

   The Open dialog box appears.

   By default, the main application option is selected and an *.IWM filter causes any matching files to appear in the FileName list box.

3. If necessary, use the Directories list box to select the directory from which you want to open the file.
4. Select a filename from the FileName list box.
5. Choose OK.

   The file is opened.

IconAuthor makes it easy to open several of the same type of files simultaneously. Follow the procedure that describes opening one file, except, at step 4, instead of selecting one filename from the Files list box, select several filenames. If you want to open all files, choose Select All.

Related Topics:

Editing between Windows

Opening Existing ASCII Text Files

When you open an existing text file, the file is placed in a new document window. You can edit the text in a text file. If two text files are open at the same time, you can edit between them. For example, you can copy text from one file and paste it into another. You can also print text files.

To open an existing file:

1. Choose Open... from the File menu..

   The Open dialog box appears.
2. Choose Text from the List Files of Type drop-down list.

   The filter in the FileName text box changes to display text file filters.
3. If necessary, use the Directories list box to select the directory from which you want to open the file.
4. Select a filename from the FileName list box.
5. Choose OK.

   The file is opened.

Related Topics:

Editing between Windows

Opening Graphic Files in the IconAuthor Work Area

When you open a graphic file through the IconAuthor File menu, the file appears in a new document window. The graphic file is available for viewing and printing only. You can move or resize the window in which the graphic is displayed, but you cannot edit the image. If you want to create and/or edit a graphic file, you must use a graphics editor. To access the IconAuthor Graphics Editor, choose Editors from the Run menu. Then choose Graphics... .

.**To open an existing file:**

1. Choose Open... from the File menu..

   The Open dialog box appears.

2. Choose Graphic from the List Files of Type drop-down list.

   The filter in the FileName text box changes to display graphic file filters.

3. If necessary, use the Directories list box to select the directory from which you want to open the file.
4. Select a filename from the FileName list box.
5. Choose OK.

   The file is opened.

Opening SmartObject Files in the IconAuthor Work Area

When you open a SmartObject file through the IconAuthor File menu, the specified page appears in a new document window. These are available for viewing only and you many print them, or move or resize the window they are displayed in. If you want to create and/or edit a SmartObject file, you must use the SmartObject Editor. To access the SmartObject Editor choose Editors from the Run menu. Then choose SmartObject...

.**To open an existing file:**

1. Choose Open... from the File menu..

   The Open dialog box appears.
2. Choose SmartObject from the List Files of Type drop-down list.

   The filter in the FileName text box changes to display the .SMT, .FTT, and .TXT file filters.
3. If necessary, use the Directories list box to select the directory from which you want to open the file.
4. Select a filename from the Files list box.
5. Choose OK.

   The file is opened in a new document window that has a Page drop-down list box at the top.
6. Use the Page drop-down list box to select the name of the page you want to view..

Saving ASCII Text Files

The first time you save a file you also name it.

Saving a file for the first time:

1. Choose Save As... from the File menu.
   The Save As dialog box appears.
2. If necessary, use the Directories list box to select the directory in which you want to save the file.
3. Enter a filename in the FileName text box.
   If you do not use an extension, IconAuthor automatically adds the default .TXT extension.
4. Choose OK.
   The dialog box is closed and the new name of the file appears in the window title bar.

Saving changes to existing files:

- Choose Save from the File menu.
  The changes are saved automatically.

Saving and Renaming Files:

It is sometimes useful to save a file and rename it at the same time. For example, you might want to change an existing text file but still want to keep a copy of it in its original state. When you save and rename the original file you have two identical files with different names. Make as many changes as you like to the copy and you will still have the original.

To save and rename a file:

1. Choose Save As... from the File menu.
   The Save As dialog box appears with the existing filename highlighted in the Filename text box.
2. Edit the filename or enter a new filename.
3. If necessary, use the Directories list box to select the directory in which you want to save the file.
4. Choose OK.
   The dialog box is closed and the new name of the file appears in the file window title bar.

Auto Saving Application Files

The Auto Save feature only applies to IconAuthor application files. It allows you to tell IconAuthor to automatically save an application, after a certain number of edits. If there is a power failure or a system malfunction while you are working with your application, a fairly up to date copy of your work will have been saved. When you begin working with IconAuthor, Auto Save is not in effect.

To turn on Auto Save:

1. Choose Auto Save... from the Options menu.

   The Auto Save dialog box appears.

2. In the Number of Edits text box, enter the number of edits after which you want IconAuthor to automatically save your work.

   An edit is any editing or building operation you perform such as building an icon into the structure or cutting an icon from the structure.

3. Choose OK.

   The dialog box is removed and Auto Save is in effect.

To turn off Auto Save:

- Perform the same procedure as is described for turning Auto Save on, however, enter 0 in the Number of Edits text box in the Auto Save dialog box.

Related Topics:

Saving Application Files

Using Page Setup

When you print a file, you can use the default margin and header/footer settings. Or, you can use the page setup procedure to customize the layout of the page. Customizing the page setup involves using the Page Setup dialog box.

To open the Page Setup dialog box:

- Choose Page Setup... from the File menu.

  The dialog box appears

Current margin settings are indicated in the four boxes to the top, bottom, left, and right of the page symbol in the Margins area. The default margin settings are 0.5".

To reset margins:

1. Click in the appropriate box in the Margins area and enter a new value.
2. Choose OK.

When you setup headers and footers, you specify the kind of information to be included and how that information will be formatted. For example, you might include today's date, and right justify it in the header. Headers and footers are always printed within the top and bottom margins, respectively. By default, the current filename appears as the header, and the word "Page" followed by a page number appears as the footer. Both items are centered on the page.

To setup headers and footers:

- Use the Header and Footer text boxes to enter text and/or any codes you want to appear at the top and bottom of each page.

  For example, you can type the word "Date" followed by a space and &d in the Header text box. This generates text such as "Date 12/15/91" at the top of the printed page. (Include spaces in your definition if you want them to appear between items.)

Use the following codes to generate and format information in the Header and Footer text boxes:

&d	Generates the current date.
&p	Generates page numbers.
&f	Generates the current filename.
&l	Justifies the text that follows at the left margin.
&r	Justifies the text that follows at the right margin.
&c	Centers the text that follows.
&t	Generates the current time.

Printing

Printing an application enables you to view its entire structure all at once. It also lets you share your design ideas with other authors. You can produce a hard copy of the structure and content of an application, and information about the variables it uses.

IconAuthor also lets you print any ASCII text or graphic file, or any SmartObject page.

To print an application file:

1. Choose Print... from the File menu.

   The Print dialog box appears
2. In the What to Print area, select the aspects of the application that you want to print.
3. In the Icon Structure Print Options area, select the way you want the structure printed.

   If you select the Range option, you must enter the coordinates of the icons that you want to print. Use the Upper left boxes to enter the X,Y coordinates of the first icon in the range. Use the Lower right boxes to enter the X,Y coordinates of the last icon in the range.

   Optionally, you can use the Zoom box to change the zoom level used for printing. By default, IconAuthor uses the current zoom level of the window being printed.

   If you want to print the coordinates of the icons, select Print Coordinates.
4. In the Icon Content Print Options area, select the way you want the content printed.

   By default, icon content is printed by column. This means that the content of icons in the first column is printed on as many pages as necessary, and a page break is automatically generated as soon as the content of icons in the second column begins printing. If you select Page break at new X, icon content is printed by row.
5. Choose OK to close the dialog box and begin printing.

To print an ASCII text file, a SmartObject page, or a graphic file:

1. Open the file and make its window the active window
2. Choose Print...

   The file is printed.

Closing Files

You can close files by double-clicking on the Control-menu boxes of the individual document windows in which the files appear. If you try to close a file that contains unsaved changes IconAuthor displays a message that the file has changed and asks if you want to save current changes. Choose Yes to save the changes, No to close without saving changes, or Cancel to stop the close process.

You can also close files by exiting IconAuthor. If you try to exit IconAuthor when one or more open files contain unsaved changes, IconAuthor displays a message asking you if you want to save changes for each document window containing unsaved edits.

When you choose Yes to save changes to a file that is being closed, the next action that IconAuthor takes depends on whether the file has previously been named. If the file has been saved before, it is saved automatically with the same name. If the file is untitled, a Save As dialog box appears.

Distributing Applications

You can distribute applications to other users who have an IconAuthor Authoring system and you can distribute them to end-users who have an IconAuthor Presentation system.

Within IconAuthor you create a resource file that keeps track of all the files used by a main application. When you are ready to distribute the application, you run IconAuthor's Resource Manager and open the application's resource file. This allows you to review all the files that are used by the application. At this point, you may choose to omit certain files from the distribution process or you may decide to include additional files, such as documentation or readme files.

During distribution, the Resource Manager can compress files and it can also split them across disks for maximum efficiency. You can also set parameters so that a setup program and Present are distributed with the application.

The Resource Manager ultimately produces a disk or set of disks that are distribution-ready. These disks contain the setup program which the end-user (or system administrator) simply runs to install the application.

Saving Application Files

As with any computer application, it is important to save your work often.

Even in the early stages of creating an application, at the point at which you have just begun to build some icons into a structure, you should save your work in an application file. *Do not* wait until you finish working to save your work.

Note: It is strongly recommended that you save an application file before you attempt to run it.

The first time you save an application you also name it.

To save an application for the first time:

1. Choose Save As... from the File menu.

   The Save As dialog box appears.
2. Enter a filename of up to eight characters in the Filename text box.

   Optionally, include an .IWM extension with the filename. If you do not use an extension, IconAuthor automatically adds the default .IWM extension.
3. Choose OK.

   The dialog box is closed and the new name of the application appears in the application window title bar.

Saving additional changes to an existing application is simpler because the application is already named.

To save additional changes to a named application:

- Choose Save from the File menu.

  The changes are saved automatically.

Related Topic:

Auto Saving Application Files

Exiting from IconAuthor

When you are ready to end your work session, you can exit IconAuthor.

To exit from IconAuthor:

- Choose Exit from the File menu.

  If you try to exit IconAuthor when multiple open files contain unsaved changes, IconAuthor displays a message such as "4 files have changed. Save current changes?" Choose Yes to make available the option to save changes, No to exit without saving changes, or Cancel to stop the exit process.

  If you choose Yes, a message appears for each file that contains unsaved changes, asking if you want to save that file. Choose Yes to save the changes, No to close without saving changes, or Cancel to stop the exit process.

  When you choose Yes to save changes to a file, the next action that IconAuthor takes depends on whether the file has been saved and named previously. If the file has been saved before, it is saved automatically with the same name. If the file is untitled, a Save As dialog box appears.

Multimedia

Your applications can include the following types of multimedia:

Audio
Animation
Graphics
SmartObjects
Text
Video

Working with Graphics

Graphic images are invaluable in making your presentations and courses more informative and more appealing. Your application can display a graphic so that it takes up a part of the screen or the entire screen. Special effects can display images in a variety of ways, such as having them appear all at once, having them fade in gradually, or having them sweep horizontally or vertically on and off the screen. Graphics can be display-only or they can be an interactive feature that the user can click on to cause an action to occur.

Create graphics for display in your applications using any graphics editor that uses the file formats supported by IconAuthor. If you need to change the colors or dimensions of a graphic, use IconAuthors graphic utility, RezSolution.

Related Topics:

Live versus Static Graphics
Displaying Live Graphics
Displaying Static Graphics
Graphic File Formats
Image Size
Image Color

Live versus Static Graphics

The role that a graphic plays is largely controlled by whether it is live or static. A live graphic (available via the SmartObject Editors Graphic object displays on the screen and can change at runtime. This is because a live graphic has properties that you can manipulate via "object" icons. Here are some examples of how live graphics can perform at runtime:

+ Use an ObjSet icon to re-set a live Graphic objects FileName property at runtime, thereby changing the graphic that it displays.
+ Set up the NotifyOnClickLeft property of a live Graphic object so that the user can click on the graphic to cause some other action to occur.
+ Set up the Drag- properties of a live Graphic object so that the user can drag and drop the graphic elsewhere on the display.

A static graphic displays on the screen and behaves as if it is part of the background. The only way to make a static graphic interactive is to use an InputMenu icon to make areas of the screen "hotspots" after the graphic is displayed. As soon as you display other information on top of a static graphic, the static graphic is replaced by the new information.

Displaying Live Graphics

Create a SmartObject file that contains a live Graphic object and then display the file with a Display icon. In addition to graphics, the SmartObject file can also display a variety of other objects such as Push Buttons, Text, List Boxes, and Movies.

Displaying Static Graphics

There are two basic ways to display static graphic files within your applications. Both methods involve the Display icon.

First, you can use a Display icon to independently display a static graphic file. The Display icon can vary the position of the graphic on the screen and vary the special effects used for display.

Second, you can include a static graphic as part of a SmartObject file. Although Graphic objects are live by default, you have the option of making them static. A static graphic has a small number of properties (characteristics) that you can set. For example, you set the FileName property to specify which file should be displayed in the object. When a Graphic object is static, its properties *cannot* change at runtime.

Graphic File Formats

The information in a graphic file is organized using one of several graphic file **formats**. The format is the specification for the structure of the file. When you use a graphics editor to create and save a file, the file is saved using one of the available formats of that particular editor.

The two primary format families are **bitmap** graphic and **vector** graphic. IconAuthor supports a variety of bitmap formats. These formats are so named because they are made up of a "map" or pattern of bits. Each bit is called a **pixel** and is, in effect, a dot, the smallest unit of a bitmap. Bitmaps tend to display more quickly than complex vector graphics but they also tend to be larger in file size because they have to describe every pixel.

IconAuthor supports the following graphic file formats:

.ATT	.CUT	.FIF	.ICO	.KFX	.PCD	.PSD	.TIF	.XPM
.BMP	.DCX	.GIF	.IFF	.LV	.PCT	.RAS	.WMF	.XWD
.CAL	.DIB	.GX2	.IMG	.MAC	.PCX	.RLE	.WPG	
.CLP	.EPS	.ICA	.JPG	.MSP	.PIC	.TGA	.XBM	

Image Size

The size of a graphic image (called the **image resolution**) is its width and height measured in pixels. As an example, a common image resolution for a graphic is 640 pixels wide by 480 pixels high. A graphic of this size would be made up of 307,200 pixels because 640 x 480 = 307,200.

Screen resolution is the total area of the computer screen and is also measured in pixels. Different systems support different screen resolutions. For example, two popular screen resolutions are 640 x 480 and 1024 x 768.

Because screen resolution can vary, you need to take certain measures to ensure that your application displays as intended on the end-users system. For example, if you design your application to run full screen in a 640 x 480 screen resolution, on a 1024 x 768 system a 640 x 480 graphic (originally intended to appear full screen) would not fill the screen.

Here are several solutions to common problems that can occur:

+ Distribute your application with multiple sets of graphics to accommodate different systems. At the beginning of the application use the System object to determine the screen resolution of the users system. Based on the results, have the application use the appropriate set of graphics. Because you are distributing a larger number of files, this method is more efficient if your application is being distributed on CD.
+ If you want the application to run fullscreen on any system, use an ObjSet icon to set the Window objects FitToWindow property to True. Any graphics that are displayed independently will be scaled to fit correctly in the window. If the application uses live Graphic objects, set their DrawStyle property to Scale or SizeByGraphic, whichever is optimal.
+ Create your graphics assuming a screen resolution of 640 x 480. Include a Window icon at the beginning of your application to cause your application to always display in a window that is 640 x 480. Use an ObjSet icon to set the Window objects FitToWindow property to True. In this case, the graphics will not scale at all.
+ Distribute your application with one set of graphics and make the system requirements clear to the end-user clear. For example, if you plan to use graphics appropriate for display in 640 x 480 screen resolution, let the user know (via packaging) that his or her system must meet these requirements.

Image Color

Each pixel in a bitmap graphic can have a color value. In a black and white bitmap, for example, black bits have a value of 0. White bits have a value of 1. In addition to monochrome, color graphics are typically 16-color, 256-color, or true color.

Similar to screen resolution, different systems use different graphics cards and they therefore support different color capabilities. For example, if a system supports 256-color graphics, it will correctly display a 16-color graphic. However, if a system supports only 16-color graphics, it will not correctly display a 256-color graphic. It will try to make the best possible color matches but the quality of the images will be reduced.

As a result, when you create or choose graphics for an application, you need to consider what kind of graphics card your end-users will have. If you are creating an application for kiosk systems where you know exactly what kind of systems will be used, you can choose your graphics correctly. However, if you are distributing a course to end-users who may have a variety of hardware configurations, you will need to plan for a greater number of possibilities.

Here are some solutions:

+ Make the requirements to the end-user clear. For example, if you plan to use 256-color graphics, let the user know (via packaging) that his or her system must support 256 colors.
+ Create the application using files that will be supported by any users system, namely, 16-color images.
+ Distribute the application with multiple sets of graphics, such as one set of 256-color graphics and another set of the same graphics in 16 colors. At the beginning of the application use the System object to determine the color support provided on the users system. Based on the results, have the application use the appropriate set of graphics. Because you are distributing a larger number of files, this method is more efficient if your application is being distributed on CD.

16-Color Graphics

16-color graphics can use up to 16 specific solid colors called the system colors. Additional colors can be created by mixing two or more colors. The mixing of colors is referred to as **dithering**. For example, pink is the result of alternating solid red and magenta pixels. These graphics do not suffer any distortion upon display because any color system will be able to display the limited number of colors correctly.

256-Color Graphics

In contrast to 16-color graphics, 256-color graphics are slightly more complex. However, they also offer much more diversity and richness. This subsection explains the basic structure of a 256-color bitmap and how to troubleshoot any problems that may arise.

Each 256-color bitmap you display has a **palette** associated with it. The palette identifies all the colors (up to 256) that are used in the image. The team member responsible for the graphics in your course must have an understanding of how palettes work. Specifically, this person needs to know how to view and edit palettes.

The role of the palette is crucial to the performance of your application. As an example, one of the most common problems that can occur is a distortion of color when two or more graphics that use different palettes are displayed side by side. The person providing your graphics must make sure that graphics that are displayed on the same frame use the same palette.

Related Topics:

256-Color Graphics and Palettes
The System Palette
Identity Palettes
Palette Troubleshooting

256-Color Graphics and Palettes

Each color in a palette has a numerical **index** from 0 to 255. The first color is index 0, the second is 1 and so on. The actual bitmap file you create contains a **color table** that lists each color index in the palette and its corresponding **RGB** (Red Green Blue) value. An RGB value is made up of three numbers (from 0 to 255) that describe the level of intensity of red, green, and blue that compose the color. For example, 0,0,0 is the RGB value for the color black which is made up of 0% intensities of red, green, and blue. White, with an RGB of 255, 255, 255 is at the opposite end of the range because it is made up of 100% intensities of red, green, and blue.

As an example, one bitmap files color table may set color 23 to RGB 0,129,0 (a shade of green) and color 24 might equal 232,0,0 (a shade of red). After the color table, the bitmap file lists each pixel and its corresponding color index. When the bitmap is displayed, the pixels are colored, in paint-by-number style, using the corresponding indexes.

The System Palette

When you work in Windows, you are using a **system palette**. The system palette has 20 reserved colors called system colors. The system colors automatically occupy the first ten and last ten positions in the palette.

When IconAuthor displays a bitmap, the bitmaps palette is realized (mapped) into the system palette. Because the first ten and last ten positions are reserved for pre-set system colors, you are actually limited to selecting 236 (rather than 256) colors when you create your bitmap. Because the first ten positions (0-9) are reserved, at realization, the first color in the bitmaps palette is mapped to position 10. The second color is mapped to position 11 and so on. Because the last ten positions are reserved, if a bitmap uses more than 236 colors, any colors at the end of the palette are lost.

Identity Palettes

An identity palette is a variation of the conventional palette that is part of a 256-color bitmap. When you make a palette into an identity palette, you pre-assign the 20 system colors to the first ten and last ten reserved positions in the palette. When the bitmaps identity palette is realized into the system palette, the first color in the bitmaps palette is mapped to position 1. The second color is mapped to position 2 and so on.

There are two basic ways to create an identity palette for a bitmap. One way is to use a graphic utility tool that has an explicit Make Identity Palette command. The command lets you automatically create an identity palette.

The other way to create an identify palette is to do so manually. That is, you can specifically fill the first ten and last ten positions with the system colors *and* make sure all of the color positions are filled. Note that you dont have to fill all positions with meaningful colors. For example, once you fill in the system colors and the custom colors used by the bitmap, you can fill the remaining positions with black.

If you are creating an identity palette manually, you will need to know the RGB values that are valid system colors. Depending on the graphic card you are using, your system is likely to support one of the two common conventions for system colors that are shown in the following table.

System Color RGB Values:

Index Number	Color Name	RGB Variation 1	RGB Variation 2
0	Black	0,0,0	0,0,0
1	Maroon	191,0,0	128,0,0
2	Green	0,191,0	0,128,0
3	Olive	191,191,0	128,128,0
4	Navy	0,0,191	0,0,128
5	Purple	191,0,191	128,0,128
6	Teal	0,191,191	0,128,128
7	Silver	192,192,192	192,192,192
8	Light Green	192,220,192	192,220,192
9	Light Blue	164,200,240	166,202,240
246	Cream	255,251,240	255,251,240
247	Medium Gray	160,160,164	160,160,164
248	Gray	128,128,128	128,128,128
249	Red	255,0,0	255,0,0
250	Lime	0,255,0	0,255,0
251	Yellow	255,255,0	255,255,0
252	Blue	0,0,255	0,0,255
253	Fuchsia	255,0,255	255,0,255
254	Aqua	0,255,255	0,255,255
255	White	255,255,255	255,255,255

Note: Dont be concerned about which set of RGB values is recognized by your graphics card. Simply choose one of the two variations in the table above and use the values consistently.

Palette Troubleshooting

The following topics describes common problems (and solutions) that can occur when using 256-color images:

Color Distortion with Side by Side Bitmaps
Color Distortion with Consecutively Display Graphics
Color Distortion with Graphics and Movies

Color Distortion with Side by Side Bitmaps

In some situations, a color distortion can occur when you display two (or more) 256-color bitmaps on the screen side by side. The first bitmap displays correctly but when the second bitmap displays (also correctly), the colors in the first bitmap are distorted. There are two likely reasons for this problem to occur.

+ First, the distortion can occur if the first bitmaps palette differs from that of the second bitmap. For example, whereas the first bitmap may use Green in position 23, the second bitmap may use Purple in position 23. This kind of color distortion is called a **palette shift**. With the display of the second bitmap, the palette values *shift* positions and are no longer accurate for the first bitmap.
+ Second, the distortion can occur if one or both of the bitmaps dont use an identity palette. When IconAuthor displays a graphic that does not have an identity palette, Windows immediately attempts to compress the palette. Colors shift positions when they are realized into the system palette and a distortion is likely to occur.

To avoid palette shift for all bitmaps that will be displayed side by side, follow two basic practices:

+ Make sure that the bitmaps use the same palette.
+ Make sure that the bitmaps all have an identity palette.

Color Distortion with Consecutively Displayed Graphics

Graphics that are displayed one after the other must also use a common palette and an identity palette. The same shift that occurs for side by side graphics will occur, for example, if you display one background graphic and then replace it with another that has a different palette.

If the graphics use different palettes, the first graphic displays correctly. When the second graphic displays, its colors are immediately distorted because Windows is still using the palette from the first graphic. As soon as the second graphics palette is realized into the Windows system palette the color distortion is corrected. Although in this kind of situation the palette shift is typically brief, it can be a distraction and a detriment to an applications performance.

The severity of this problem depends on the speed of the machine on which the application is running. If you are using a slower machine, the distortion will be prominent. If you are using a relatively fast machine, the distortion may happen so quickly that it is not discernible to you. Because most developers do not have complete control over the speed of the end-users machine speed, remember to use a common palette that is an identity palette for all the graphics in a course.

Color Distortion with Graphics and Movies

The SmartObject Editor lets you use the Movie object to include movies in your application. Be aware that when the application plays a movie and accompanying graphics must use only the 20 system colors.

Movies use the 20 system colors and up to 236 other colors. If you try to display a graphic that uses colors other than those used by the movie, a palette shift will occur.

True Color Graphics

The two basic advantages to using true color graphics are their photographic quality and the fact that they do not cause any of the color distortions that can occur with 256-color graphics. However, the files are very large (three times the size of 256-color graphics) and therefore take up more storage space and are slower to display. Also, because true color represents the latest in color technology, end-users are less likely to have the necessary graphics card.

Working with Animation

Animations bring exciting and informative motion to your applications. Your application can play an animation that takes up a part of the screen or the entire screen. There are several ways to create animations and a variety of ways to play them.

One way to create animations is to use IconAnimate, IconAuthors animation editor. You can then run the animation in the following ways:

+ Play the animation in a live IconAnimate object available through the SmartObject Editor.
+ Play the animation using the Display icon.

The other way to create animations is to use a third-party tool such as AutoDesk Animator. Once you create the file you can play it in a live Movie object available through the SmartObject Editor.

Working with Text

Text is one of most basic tools you can use in your visual displays. Like a graphic, it can be generated from a file or it can be generated at runtime. You can display anything from a single character to a full screen of text. If you need to display more text than will fit on the screen at once, you can display the text in a scrollable box or you can change the text when the user is ready to move on.

Text can be display-only or it can be interactive. There are several basic ways in which text can be interactive. One common situation is where text is intended for the user to edit. Or, in other cases, a rectangular area of text can serve as a button that a user can click on to cause some other action to occur.

Related Topics:

Live versus Static Text
Displaying Live Text
Displaying Static Text
Creating Text Files

Live Versus Static Text

The role that text plays is largely controlled by whether it is live or static. Live text (available via the SmartObject Editors Text object) displays on the screen and can change at runtime. This is because live text has properties that you can manipulate via "object" icons. Here are some examples of how live text can perform at runtime:

+ Use an ObjSet icon to re-set a live Text objects FileName property at runtime, thereby changing the text file that it displays.
+ Set up the NotifyOnClickLeft property of a live Text object so that the user can click anywhere on the text to cause some other action to occur.
+ Include one or more Hotwords (interactive words) in a Text object. When the user clicks on a Hotword, you can cause a specific action to occur based on the selected word.
+ Use a live Text object to let the user browse database records at runtime.

In contrast, static text displays on the screen and behaves as if it is part of the background. There are several ways to display static text such as via a static Text object, the Write icon, or the Text icon. The only way to make static text interactive is to use an InputMenu icon to make areas of the screen "hotspots" after the text is displayed. As soon as you display other information on top of a static text, the text is replaced by the new information.

Displaying Live Text

You can create a SmartObject file that contains a live Text object and then display the file with a Display icon. When you use live objects you can change their properties at runtime via object icons.

There are three ways to enter text in a live Text object. First, you (the author) can type in it within the SmartObject Editor. You set several characteristics of the text such as formatting, line spacing, and paragraph alignment. If there is more text than can be displayed at one time, you can make the text scrollable.

Second, you can set the object's FileName property to the particular file you want it to display. When you do this you also need indicate whether the text file should be embedded in or linked to the SmartObject file. When you embed a file, it has the same effect as pasting the text into the file. When you link a text file, it is still a separate file that is called by the SmartObject file at runtime.

Third, you can make the object editable (by setting its Editable property to True) and thereby allow the user to type in the object at runtime.

In addition to allowing user input, because the Text object is live, there are other ways in which it can perform interactively. For example, it can be set up to cause execution to flow down a particular branch when the user finishes providing input. Or, the object can be set up to act as a button, which when clicked upon, causes some other action to occur.

Displaying Static Text

Use the following basic techniques to display static text within your applications:

Use a static Text object in a SmartObject file
Use a Write icon
Use a Text icon
Use an Input Icon

Static Text Objects

The SmartObject Editor lets you use a Text object to display static text. Like other available objects such as the Graphic object and Push Button, the Text object is live by default. As necessary, you must explicitly make it static.

There are two ways to enter text in a static Text object. You can type in it, setting several characteristics such as formatting, line spacing, and paragraph alignment. Or, you can specify the name of a text file that will serve as the contents of the object. The text file can be formatted (with different color and font information) or it can be an unformatted ASCII file.

A static Text object has a small number of properties (characteristics) that you can set. For example, you can use the FileName property to specify the particular text file you want to use. You can also set the object's SelectionArea property if you plan to make the object a hotspot at runtime (with an InputMenu icon). When a Text object is static, its properties *cannot* change at runtime.

Static Text via the Write Icon

One common way to display up to one line of text is to use the Write icon. You specify the text string (or a variable that contains a string) within the Content Editor and the string is displayed at runtime.

Precede the Write icon with a Color icon to control the color of text. If you do not use a Color icon, IconAuthor uses the system default values for black text on a white background. Precede the Write icon with a Fonts icon to control the font style and size used for text.

Static Text Files via the Text Icon

Use a Text icon to display a static unformatted text file. The user cannot interact with text displayed via the Text icon. You can create an ASCII file with an IconAuthor Text window, the SmartObject Editor, or a text editor such as Notepad. Use the Font icon to control the overall appearance of text displayed with the Text icon. By default, if you do not use the Fonts icon, the smallest available size of the System type font is in effect for text displays. Text is black by default. Precede the Text icon with a Color icon to change the color of text.

Displaying Static User Input: The Input Icon

If your application does not use live objects, use the Input icon to allow the user to input different types of information, including text, numbers, or a date. The Content Editor of the Input icon lets you access the Input Selection Editor. This editor allows you to specify exactly how and where you want the user's input to appear on the screen. For example, you can specify that input is displayed on the screen exactly as it is entered or that it is masked by a character for security purposes. (You can also choose not to have input appear on the screen at all.)

Like the Write icon, the style, size and color of text generated through the Input icon are controlled by the Font and Color icons.

Creating Text Files

The following table outlines the different ways to create text files for your IconAuthor applications.

Editors:	File Formats:	Display Options in IconAuthor:
SmartObject Editor - Formatted Text mode	.FTT	Use the SmartObject Editors Text object.
SmartObject Editor - ASCII Text mode	.TXT	Use the SmartObject Editors Text object or a Text icon.
IconAuthor Text window	.TXT	Use the SmartObject Editors Text object or a Text icon.
External ASCII Text Editor such as Notepad	.TXT	Use the SmartObject Editors Text object or a Text icon.
External Word Processor	.RTF	Open the file in the SmartObject Editor and save as .FTT. (Rudimentary .RTF formatting such as color and font is supported. Complex formatting such as tables, columns, and embedded graphics is not.) Link to a SmartObject Editor Text object.

Working with Audio

Audio is one of the most effective presentation techniques you can use. It can accompany and enhance any one of the other media you are including in your application such as graphics, video, and animation. Audio is a unique and powerful way of involving the user in the application. You can use voice, high quality musical recordings, and special effects to make your applications more appealing and more informative.

In an application created for educational purposes, audio can provide valuable voice instruction as well as feedback in test situations. For example, audio can describe what is going on in an instructional video or graphic display. Or, a user's correct answer can be responded to with a voice message such as "That answer is correct. Please press any key to move on to the next question." Also, audio is a key part of multimedia language lesson applications that teach skills such as listening, reading, and speaking.

In a presentation or kiosk environment, voice audio is also useful to enhance and describe the visual information. Music and special effects play a crucial role in making the application richer, more enjoyable, and even more alluring. Remember, even if a user is not yet viewing the monitor, effective audio can be a powerful way of attracting one's attention.

Related Topics:

How Audio Works
Supported File Formats
The Display Icon and Audio Objects
MCI Icons
Movies that use Audio
Beeps

How Audio Works

Although adding audio to your application can be as simple as drawing and defining an Audio object in the SmartObject Editor, it is important that you understand the power behind the audio.

IconAuthor applications play audio files through the Media Control Interface (MCI). MCI is a feature that comes with your windowing software. In effect, MCI is a channel of communication between IconAuthor and an audio device. Any device that supports MCI can be manipulated by your applications.

To play audio files, your system (and your end-users systems) must be equipped with an MCI-compatible sound card (such as the SoundBlaster). You can also use a CD-ROM drive to play sounds. Note that in order to hear the audio, systems should also be equipped with speakers or headphones.

Audio File Formats

You can include two types of audio files in your applications: **wave audio** and **MIDI (Musical Instrument Digital Interface)**. Wave audio (.wav) files are well-suited for voice recordings such as a person welcoming the user to the course or providing feedback by saying That is correct. MIDI (.mid) files play music or special audio effects.

Sometimes the sounds you want to play are contained in discrete files. For example, That is correct is in correct.wav and That is incorrect is in WRONG.WAV. However, in other situations, one file may contain multiple sounds.

Be sure to inform the person who creates your sound files that it is *easiest* if each sound clip is contained in a unique file. If a file contains just one sound clip you will only have to specify to play the file from beginning to end. Otherwise, you will have to specify the precise starting and ending positions (in seconds) of the portion of the file that you want to play.

The Display Icon and Audio Objects

The simplest way to play audio is to create SmartObject files that play Audio objects when displayed via the Display icon. Audio object can play wave audio files, MIDI files, or CD audio selections. These features are available because, behind the scenes, IconAuthor is sending commands to MCI. When you use the Audio object you do not have to explicitly construct MCI commands. You simply set properties for the object. IconAuthor takes care of the commands for you.

The Audio object you use can perform in one of two ways. It can be invisible but play automatically when a Play command is issued to it. Or, you can set up a Button object to control the audio. To have the audio object be invisible but play automatically, set its CommandOnCreation property to Play. It will play audio as soon as the SmartObject page displays. Otherwise, at some other point after display, you can make the object play by using an ObjSet icon to set the objects Command property to Play.

To control the audio via a Button object, set the Button objects ControlObjectName property to the ObjectName of the audio object and its ControlCommand property to the appropriate audio command. The user will then be able to click on the Button object to play the audio.

MCI Icons

The MCI icon also allows your IconAuthor applications to take advantage of the enhanced audio features (and any other multimedia features) of MCI. Each MCI icon can send a different command to a supported sound card or CD-ROM drive. Therefore unlike the Audio object, if you plan to use MCI icons, you need to be familiar with the MCI commands and syntax.

Note: Most authors will find that the Audio object is not only easier to use than MCI icons, but is also versatile enough to fulfill all of their audio needs.

Although the MCI icon itself does not contain any default values, IconAuthor comes with three composites that make learning about MCI easier. These composites, called CD-Audio, MIDI, and WaveAudio, are primarily made up of MCI icons. The MCI icons that comprise these composites already contain commands that require little or no editing to be ready for play. For example, if you build a WaveAudio composite into your structure you only have to specify the path and filename of the wave audio file you want to play.

Movies with Audio

The visual images provided by digital or analog video are often accompanied by audio. For example, when you use the Movie object to play a Video For Windows (.AVI) file, it frequently plays audio as well. The same is true if your application uses Video icons to play video from a videodisc player.

Beeps

Use the Beep icon when you want a system to emit a simple low or high tone. You can use one Beep icon to create a single tone, or you can create a series of beeps, by using several Beep icons (or a Beep icon within a loop). For example, a beep can be useful in notifying the user that a response is correct (a high tone) or incorrect (a low tone). Note that the Beep icon does not require any additional hardware.

Working with Video

The easiest way to play video in your IconAuthor applications is through the Movie object.

The Movie Object

Although adding video to your application can be as simple as drawing and defining a Movie object in the SmartObject Editor, it is important that you understand how video works

The Movie object plays video files through the Media Control Interface (MCI). MCI is a feature that comes with your windowing software. In effect, MCI is a channel of communication between IconAuthor and a video device and/or video software. Any device that supports MCI can be manipulated by your applications.

Prerequisites

To play video files, your system (and your end-users systems) must be equipped with MCI-compatible video software and/or hardware. Some MCI video support is software-only. That is, you and your end-users only need special software, not hardware. An example of this is the Video For Windows driver for MCI which comes with IconAuthor. Some other kinds of MCI video support require special software and hardware.

Note: If you plan to play video files that use sound, your system also requires a sound card that supports MCI.

Related Topic:

Playing the Movie Object

Playing the Movie Object

The simplest way to play video is to create a SmartObject file that plays a Movie object when displayed via the Display icon. When you play a Movie object, behind the scenes IconAuthor is sending commands to MCI. You do not have to explicitly construct MCI commands. You simply set properties for the object. IconAuthor takes care of the commands for you.

The Movie object can perform in one of three ways. If you set the objects ControlBar property to True, it appears with controls that allow the user to play the video at will. The second way to play a movie is to explicitly set the object to play. If you set the objects CommandOnCreation property to Play, it will play the video as soon as the SmartObject page displays. Otherwise, at some other point after display, you can make the object play by using an ObjSet icon to set the objects Command property to Play. The third way to play the movie object is via a Button object. You can set up a Push Button or Picture Push Button style Button object to control the playing of a movie. Set the ControlObjectName property of the button object to the ObjectName of the movie object. Also, set the ControlCommand property to the appropriate movie command. The user will then be able to click on the button to play the movie.

Understanding IconAuthor Path Files

By default, IconAuthor uses a main directory and subdirectories that are set up during installation. IconAuthor uses the subdirectories to open and save files. It expects to find specific file types in each subdirectory. Files IconAuthor looks for are:

Subdirectory	File Type
ANIMATE	IconAnimate Animation
AUDIO	Audio
DATABASE	Database format
FORMAT	Format
GRAPHICS	Graphic
HELP	Help
ICONWARE	Main and Sub-Application
INPUT	Input menu template
MOVIE	Digital Video and third-party animations
LIBRARY	Icon library
TEXT	SmartObject, ASCII
VARIABLE	Variable

IconAuthor has a special path file that contains the names of the subdirectories and the extensions of the files that are stored there. The path file is in the IAUTHOR directory and is called IAUTHOR.PTH. IAUTHOR.PTH contains the following path variables and values:

@_ANIMATE_PATH	C:\IAUTHOR\IANIMATE
@_GRAPHIC_PATH	C:\IAUTHOR\GRAPHICS
@_ICONWARE_PATH	C:\IAUTHOR\ICONWARE
@_VARIABLE_PATH	C:\IAUTHOR\VARIABLE
@_TEXT_PATH	C:\IAUTHOR\TEXT
@_FORMAT_PATH	C:\IAUTHOR\FORMAT
@_DATA_PATH	C:\IAUTHOR\DATABASE
@_LIBRARY_PATH	C:\IAUTHOR\LIBRARY
@_MOVIE_PATH	C:\IAUTHOR\MOVIE
@_USER_PATH	C:\IAUTHOR\USER
@_INPUT_PATH	C:\IAUTHOR\INPUT

When you create and save applications, IconAuthor creates a unique path file for each application. IAUTHOR.PTH is a master that is used to create the application path file. The first time you save an application file, a path file of the same name is automatically created with a .PTH extension. The path file is saved in the same place as the application file. For example, when you save DEMO1.IWM, IconAuthor automatically creates DEMO1.PTH.

A path file is a list of path variables. Path variables are always assigned values that are paths. For example, if you installed IconAuthor on the C drive, the path variable called @_GRAPHICS_PATH is assigned the value C:\IAUTHOR\GRAPHICS.

Related Topic:

Customizing the Directory Structure
Path Variables

Customizing the Directory Structure

Advanced computer users may want to change the default directory structure. Instead of storing files the way IconAuthor is originally set up to store them, you may want to store all files for an application in the same directory. Or, you may want to have a different set of directories for each application you create. You can create a custom directory structure by changing the path file.

Before changing the path information, use a utility such as DOS or the Windows File Manager to create the directories and subdirectories. Then edit the .PTH file of your application file to reflect the new directories and subdirectories you just created.

To edit the .PTH file of an application:

1. Open the application.
2. Choose Window Contents from the View menu.
3. Choose Path Variables.

   The contents of the application window changes to reflect the path variables and values in the application's path file.
4. Select the path you want to change.
5. Type a new path in the box at the top of the window.
6. Press RETURN.

   The new path value is assigned to the appropriate variable. Continue to change path values as necessary.

Hint: You can easily set multiple path variables to the same path. While pressing and holding the CTRL key, click on the items you want to reset. Type the new value and press RETURN. Or, drag across consecutive items to select them.

You can also make changes to a .PTH file by using the information in another existing .PTH file.

To use another .PTH file to change .PTH settings:

1. Open the application whose .PTH you want to change.
2. Choose Variables from the Edit menu.
3. Choose Set Path From File...

   The window where your application structure was visible changes to show the corresponding path variables. A File Open dialog box appears.
4. Double click on the .PTH file you want to use.

   The path variable values for the current application are changed to match the path variable values of the file you chose.

When you save the application file, the changes to the path file are automatically saved. To change the view back to the application structure, choose Window Contents from the View menu again, and choose Application Structure.

If you do not plan to use IconAuthor's default directory structure, you can edit the master IAUTHOR.PTH file. If you change IAUTHOR.PTH, the new IAUTHOR.PTH file will be used to create the path file for each subsequent application that you save. IAUTHOR.PTH is an ASCII text file that can be edited in Notepad or in an IconAuthor text window.

Warning: Do not change the name of the IAUTHOR.PTH file or any of the application path files.

IconAuthor Index

The following Help topics are available for IconAuthor. Use the scrollbar to see entries that are not currently visible.

Circle Icon
Clear Icon
Color Icon
DDE Icon
DllCall Icon
Dll Link Icon
Database Icon
Date&Time Icon
Display Icon
Ellipse Icon
Exit Icon
Font Icon
Help Icon
If Icon
Input Icon
InputMenu Icon
Line Icon
LoadVar Icon
Loop Icon
LoopIndex Icon
MCI Icon
Menu Icon
MIDI Icon
Module Icon
MsgBox Icon
Note Icon
ObjDelete Icon
ObjEvent Icon
ObjGet Icon
ObjMenu Icon
ObjQueue Icon
ObjSet Icon
Parse Icon
Pause Icon
Print Icon
Program Icon
RS-232 Icon
Random Icon
SaveVar Icon
Shuffle Icon
Snapshot Icon
Startup Icon
SubApp Icon
SubAssign Icon
Subroutine Icon
Text Icon
V:Audio Icon
V: Frame #? Icon
V: Overlay Icon
V:PlayTo Icon
V:Player Icon
V:Segment Icon
V:Still Icon
Variable Icon
WaveAudio Icon
Window Icon

Write Icon

IconAuthor Keys

Use the following accelerator keys in IconAuthor:

Menu	Command	Key(s)
File	Save	Ctrl + S
Edit	Cut	Shift + Del
	Copy	Ctrl + Insert
	Paste	Shift + Insert
	Clear	Del
	Find...	Ctrl + F
	Find Next	F3
Edit / Application	Compress Composite	Ctrl + C
	Make Composite	Ctrl + O
	Disable Selection	Ctrl + D
	Enable Selection	Ctrl + E
	Add Content...	Ctrl + A
Run	Application From Top	Ctrl + R
Window	Tile	Shift + F4
	Cascade	Shift + F5

Additionally, you can use the following keys to perform the corresponding functions:

Key	Function
Escape	1) To cancel a File Open process, press Escape. 2) To stop an application from running, press Escape.
Enter	When an application window is active, press Enter to open the Content Editor of the selected icon.
F4	When a Content Editor is open and the cursor is in a text box that has a corresponding drop-down list box, press F4 to drop-down the list.
Page Up	Scrolls up, one screen at a time, in the active window. Valid for any window in the work area. Also valid for the Icon Library.
Page Down	Scrolls down, one screen at a time, in the active document window. Valid for any window in the work area. Also valid for the Icon Library.
Ctrl + Page Up	Scrolls to the left, one page at a time, in the active document window. Valid for any window in the work area.
Ctrl + Page Down	Scrolls to the right, one page at a time, in the active document window. Valid for any window in the work area.
Home	1) Jumps to the top of an active document window that contains an application, a graphic, or a SmartObject page. Also valid for the Icon Library. 2) Jumps to the beginning of the current line in a text window.

End	1) Jumps to the bottom of an active document window that contains an application, a graphic, or a SmartObject page. Also valid for the Icon Library. 2) Jumps to the end of the current line in a text window.
Ctrl + Home	Jumps to the left of an active document window.
Ctrl + End	Jumps to the right of an active document window.
Control	Press the Control key before starting to drag an icon in the structure, to copy rather than move it.
Any Letter Key	When focus is in the Icon Library, press any key to jump to the first icon that begins with that letter. Press the same key again to jump to the next icon that begins with that letter.

IconAuthor Commands

To get help with a command, choose the appropriate menu.

File Menu

New...
Open...
Save
Save As...
Properties
    Register Resources
    File Type...
Library
    Open...
    Save
    Save As...
Delete...
Page Setup...
Print...
Printer Setup...
Exit

Edit Menu

Cut
Copy
Paste
Clear
Select All
Copy To...
Paste From...
Find...
Find Next
Application
    Compress Composite
    Make Composite...
    Add Composite to Library
    Disable Selection
    Enable Selection
    Enable All

- Add Content...
- Library
  - Expand Categories
  - Compress Categories
  - Add Content...
  - Build
  - Remove Icon
- Variables
  - Clear Application Variables
  - Set Path From File...

Run Menu

- Application From Top
- Application From Selected
- Debug
  - Set Stop Point
  - Clear Stop Point
  - Clear All Stop Points
- Default Windows Setup...
- Editors
  - Animation...
  - Graphics...
  - IAScope...
  - Resource Manager...
  - RezSolution...
  - SmartObject...
  - Video...
  - Calculator
  - Clipboard
  - Notepad

Options Menu

- Structure Setup...
- Library Setup...
- Color Scheme...
- Auto Save...
- Video Setup...
- Overlay Setup...
- Audio Setup...
- Add Content On Build
- Backup Structure On Save
- Confirm Clear

View Menu

- Library
- Ribbon
- Status
- Zoom
  - 25%
  - 50%
  - 75%
  - 100%
- Window Contents
  - Structure
  - User Variables

System Variables
Path Variables

Window Menu

Tile
Cascade
Close All
Duplicate
Show File Path

Help Menu

Index
Keyboard
Commands
Procedures
Using Help
About IconAuthor...

IconAuthor Procedures

Building

Icons
Building Structures
Finding Icons in the Library
Jumping to an Icon in the Library
Dragging and Dropping Icons
Composite Icons

Adding Content

Adding Content
Content Editors
Icon Colors
Naming Icons
Entering Values
Types of Values for Content Editor Fields

Editing Applications

Editing Applications
Editing Structures
Selecting Icons
Cutting Icons
Copying Icons
Pasting Icons
Clearing Icons
Editing Between Windows

Running Applications

Running an Application
General Execution Rules
Disabling and Enabling Icons
Debugging

Variables

Variables
User Variables
Indexed Variables
System Variables
Path Variables

Multimedia

Audio
Animation
Graphics
SmartObjects
Text
Video

Object Properties

Audio
Button
Combo Box
Database

Graphic
IconAnimate
Keyboard
List Box
Menu
Movie
OLE
System
Text
Timer
Transparent
Variable
Window

Managing Application Files

Organizing Files
Starting Applications
Naming Files
Saving Files
AutoSaving
Opening Applications
Closing Files
Exiting from IconAuthor
Distributing Applications
Opening Graphic Files in the IconAuthor Work Area
Opening SmartObject Pages in the IconAuthor Work Area
Path Files
Using Page Setup
Printing

Managing ASCII Text Files

Starting ASCII Text Files
Naming Files
Opening ASCII Text Files
Closing Files
Saving ASCII Text Files
Using Page Setup
Printing

Editing ASCII Text Files

Editing ASCII Text Files
Editing between Windows

File Menu

New... command

Starts a new application or text window.

Related Topics:

Starting New Applications
Starting to Create ASCII Text Files

Open... command

Opens a previously created application file, graphic file, ASCII text, or SmartObject file.

Related Topics:

Opening Existing Applications
Opening Existing ASCII Text Files
Opening Graphic Files in the IconAuthor Work Area
Opening SmartObject Files in the IconAuthor Work Area

Save command

Saves changes to an existing application file.

Related Topics:

Saving Applications
Saving ASCII Text Files

Save As... command

Saves and names an application file for the first time. Or saves and renames an existing application file.

Related Topics:

Saving Applications
Saving ASCII Text Files

Properties

Displays a cascading menu of commands specific to preparing files for distribution via the Resource Manager.

> **Register Resources**
>
> Allows you to register resources for an application.
>
> **File Type...**
>
> Allows you to specify whether a file is a main application or a sub-application.

Library command

Displays a cascading menu of File menu commands that apply specifically to library files.

> **Open... command**
>
> Loads a different library file.
>
> **Save command**

Saves changes to an existing library file.

Save As... command

Saves and names a library file for the first time. Or saves and renames an existing library file.

Delete...

Deletes an application, graphic, ASCII text, or SmartObject file.

Page Setup...

Sets the top, bottom, left, and right margins and specifies information to be included in the header and footer of a printed page.

Related Topic:
Using Page Setup

Print...

Print an application, graphic, ASCII text, or SmartObject page.

Related Topic:
Printing

Printer Setup...

Selects a printer driver and a printer connection

Exit

Exits IconAuthor. Gives you the opportunity to save changes to application and ASCII text files.

Edit Menu

Cut command

Deletes the selected icon(s) from an application, or deletes the selected text from a text window. The cut item is placed on the Clipboard.

Related Topics:

Cutting Icons
Editing ASCII Text Files

Copy command

Copies the selected icon(s) from an application, or copies the selected text from a text window. The item is placed on the Clipboard.

Related Topics:

Copying Icons
Editing ASCII Text Files

Paste command

In an application file, causes the cursor to appear as the Clipboard symbol. The Clipboard can then be dragged and dropped to a position in the structure. In a text window, inserts the text on the Clipboard at the current insertion point.

Related Topics:

Pasting Icons
Editing ASCII Text Files

Clear command

Clears the selected icon(s) from a structure or clears the selected text from a text window. Cleared items are permanently removed and are not placed on the Clipboard.

Related Topics:

Clearing Icons
Editing ASCII Text Files

Select All command

In an application, selects every icon in the structure. In a text window, selects all the text.

Related Topic:

Selecting Icons

Copy To... command

Copies selected icon or icons to a new application file.

Related Topic:

Copying Part of a Structure to a New File

Paste From... command

Allows you to insert the structure of another application into the structure in the active application window.

Related Topic:
Inserting Another Application into the Current Structure

Find... command

Locates an icon, of a particular name, in the structure

Related Topic:
Jumping to Icons in a Large Structure

Find Next command

Locates the next icon, of a particular name, in the structure. Used after the Find... command.

Related Topic:
Jumping to Icons in a Large Structure

Application command

Displays a cascading menu of Edit menu commands that apply only to application files.

Compress Composite command

Hides all the icons in a composite, except the lead icon.

Related Topic:
Compressing Icons in a Composite

Make Composite... command

Makes a selected range of icons into a composite.

Add Composite to Library command

Adds a composite to the Custom folder of the icon library.

Disable Selection command

Makes a selected icon or a selected range of icons temporarily un-executable.

Related Topic:
Disabling and Enabling Icons

Enable Selection command

Makes a selected icon or a selected range of icons executable. Undoes the Disable Selection command.

Related Topic:
Disabling and Enabling Icons

Enable All command

Enables all icons in a structure. Undoes the Disable Selection command.

Related Topic:
Disabling and Enabling Icons

Add Content... command

Opens the Content Editor of a selected icon in the structure.

Related Topic:

Opening Content Editors

Library command

Displays a cascading menu of Edit menu commands that apply specifically to library files.

Expand Categories command

Opens all library folders.

Compress Categories command

Closes all library folders.

Add Content...command

Opens the Content Editor of a selected icon in the icon library.

Build command

Causes the cursor to appear as the currently selected icon in the library. Move the cursor to a position in the structure and click to build.

Related Topic:

Building Structures

Remove Icon command

Removes the selected icon from the library.

Variables command

Displays a cascading menu of Edit menu commands that apply specifically to variables.

Clear Application Variables command

Clears any application variables previously generated when an application was run.

Set Path From File...command

Allows you to use another .PTH file to reset the path variable values in the .PTH file that corresponds to the current application.

Related Topics:

Understanding IconAuthor Path Files
Customizing the Directory Structure

Run Menu

Application From Top command

Runs all enabled icons in the application from the Start icon.

Related Topics:

Running an Application
General Execution Rules
Disabling and Enabling Icons

Application From Selected command

Runs all the enabled icons in the application from a selected icon.

Related Topics:

Running an Application
General Execution Rules
Disabling and Enabling Icons

Debug command

Displays a cascading menu of commands for debugging applications using IAScope.

Set Stop Point... command

Sets debug stop points.

Clear Stop Point command

Clears currently selected stop point.

Clear All Stop Points command

Clears all stop points.

Default Windows Setup... command

Designates the way in which an application appears when executed, for example, in a resizable window, or full screen.

Related Topics:

Default Window Setup

Editors command

Displays a cascading menu of commands that run various IconAuthor editors and other external programs.

Animation... command

Runs IconAnimate, the IconAuthor animation editor. Lets you create and save animation files.

IAScope... command

Runs the IAScope, the IconAuthor visual debugging program. Lets you debug your IconAuthor application.

Resource Manager...

Runs the IconAuthor Resource Manager. Lets you organize and distribute your applications.

RezSolution... command

Runs the IconAuthor graphics utility, RezSolution. Lets you alter the resolution of graphics.

SmartObject... command

Runs the IconAuthor object-oriented text editor. Lets you create SmartObject files.

Video... command

Runs the IconAuthor Video Editor. Lets you view material from a videodisc, videotape, or digital video file.

Calculator

Runs the Calculator.

Clipboard

Runs (shows the contents of) the Clipboard.

Notepad

Runs Notepad, the text editor.

Options Menu

Structure Setup... command

Allows for customizing the characteristics of icons in the structure. For example, icons can be displayed with or without labels.

Library Setup... command

Allows for customizing how icons appear in the library. For example, folders can be open or closed.

Color Scheme... command

Allows for customizing the colors used to indicate the current state of icons. For example, whether they are selected.

Auto Save...

Accesses the Autosave feature which can be set to automatically save application files after a specified number of edits have been performed. By default, this feature is turned off.

Video Setup... command

Allows for setup of a specific videoplayer. For more information, see the IconAuthor Getting Started book.

Overlay Setup... command

Allows for setup of a specific video overlay card. For more information, see the IconAuthor Getting Started book.

Audio Setup... command

Allows for setup of a specific audio card. For more information, see the IconAuthor Getting Started book.

Add Content On Build

When this option is toggled on the Content Editor will be automatically opened each time an icon is added to the structure.

Related Topic:

Adding Content

Backup Structure On Save command

When this option is toggled on, IconAuthor creates a backup file of the application file currently being save. The backup file has a .BAK extension.

Confirm Clear command

Causes IconAuthor to ask you to confirm whether you want to clear icons when you use the Clear command from the Edit menu or drag an icon to the trashcan.

Related Topic:

Clearing Icons

View Menu

Library command

When this option is toggled on, the Icon Library is visible.

Ribbon command

When this option is toggled on, the ribbon bar is visible.

Status command

When this option is toggled on, the status bar is visible.

Zoom command

Displays a cascading menu of commands that zoom or un-zoom the structure.

Related Topic:
Zooming the Structure

25% command

Zooms the structure to 25%.

50% command

Zooms the structure to 50%.

75% command

Zooms the structure to 75%.

100% command

Zooms the structure to 100%.

Window Contents command

Displays a cascading menu of commands that change the current view in a active application window.

Structure command

Displays the structure of an application.

User Variables command

Displays the user variables associated with an application.

System Variables command

Displays the system variables associated with an application.

Path Variables command

Displays the path variables associated with an application.

Window Menu

Tile command

Rearranges the currently open document windows so that all are visible within the work space.

Cascade command

Causes the currently open document windows to overlap so that each title bar is visible.

Close All command

Closes all of the currently open document windows. Asks if you want to save a file if it contains unsaved changes.

Related Topic:

Closing Files

Duplicate command

Makes a duplicate of the active document window. If you make changes to information in a duplicate window, the changes are also made to the original and any other duplicates of that window.

Related Topics:

Editing Between Windows

Show File Path command

When this option is toggled on, full paths are displayed for document window title bars and minimized document windows.

Help Menu

Index command

Displays a list of Help topics.

Keyboard command

Displays a table of accelerators for performing some tasks with the keyboard instead of the mouse.

Commands command

Displays an explanation of commands.

Procedures command

Displays a description of how to use IconAuthor.

Using Help command

Displays a short tutorial and other information about using online Help.

About IconAuthor... command

Displays IconAuthor copyright and version information.

Accelerator Property

This property is only available via the Menu object's Menu Design dialog box within the SmartObject Editor. It sets the accelerator key for a Menu Item.

AlignHorizontal Property

Resets the horizontal alignment of all text in a Text object. Set this property to Left, Right, or Center. This property is available at runtime only.

Alignment Property

If the AutoTrack property is True, the Alignment property sets the position where a pop-up menu appears relative to the cursor position. Set this property to Left, Right, or Center. For example, if you set the property to Left, when the pop-up menu appears, the cursor is on its left. (This property is available for Windows 3.1 only.)

Area Property

Resets the area of the object. This property is available at runtime only. Specify four numbers separated by commas to describe the screen coordinates of the object's upper left corner and the object's width and height.

Example: 100,100,50,50 specifies that the object's upper left corner is at 100,100 and the object is 50 x 50 pixels.

AutoErrorDisplay Property

This property, set to True or False, suppresses error messages from the ODBC driver. If the property is set to True, error messages from the ODBC driver will display as necessary. If the property is set to False, all error messages from the ODBC driver will be suppressed.

AutoNavigate Property

This property, set to True or False, controls whether the control bar buttons let the user navigate through the Database records. If True, the control bar buttons are automatically set up to allow record navigation. If False, the control bar buttons will not allow record navigation unless you set the Notify- commands to True.

AutoTrack Property

This property, set to True or False, controls where a pop-up menu appears. If True, the menu appears where the mouse click occurred. If False, the menu appears where you positioned the Menu object on the SmartObject page.

BaseLine Property

This property, set to True or False, controls whether an underline appears in the object to indicate how many spaces are available for input.

Example: If an object has an input style of Currency, BaseLine is True, and InputLimit is set to 10, the underline will extend out to 10 positions to show the user how many digits can be specified.

Border Property

This property, set to True or False, controls whether a border appears around the graphic.

Border Width Property

This property lets you control the width of the border, in pixels, around the graphic. The default is 1.

Bottom Property

This property is the distance (in pixels) from the top of the page to the bottom of the object. Example: 100.

This property can only be manipulated at runtime via the IconAuthor object icons.

ButtonStates Property

This property lets you choose how you many states your Picture Push Button will have. The choices are: Up (1 state), UpDown (2 states) or UpDownDisabled (3 states.) Your graphic file needs to have the corresponding number of graphics in it.

CanAppend Property

This property, set to True or False, controls whether the user can add new records to the recordset.

CanPlayCD Property

This property, returning True or False, indicates whether the system IconAuthor or Present is running on can play CD selections.

CanPlayMIDI Property

This property, returning True or False, indicates whether the system IconAuthor or Present is running on can play MIDI files.

CanPlayMovie Property

This property, returning True or False, indicates whether the system IconAuthor or Present is running on can play digital video and animation files.

CanPlaySound Property

This property, returning True or False, indicates whether the system IconAuthor or Present is running on can play sound.

CanPlayVideo Property

This property, returning True or False, indicates whether the system IconAuthor or Present is running on can play analog video files.

CanUpdate Property

This property, set to True or False, controls whether the recordset can be updated by the user.

Caption Property

This property resets the menu item appearance within a menu. Type a simple command name such as "Copy" or "Change Color..." Or use one of the following special formatting characters.

&	Preceding a character with the "&" symbol causes that character to appear underlined. Following convention, if a Menu Item that is a menu heading (in top level menu) contains an underlined character, the user can press and hold the Alt key and type the underlined character to select the menu. Also, if a Menu Item that is a command contains an underlined character, the user can select the menu and then press the underlined character to choose the command. For example: &File generates "File".
\t	These characters generate a Tab within the caption. A Tab typically separates the name of a command from an accelerator key combination. For example: &Cut\tCtrl+C generates "Cut Ctrl+C"
\a	These characters cause any characters that follow to be right justified on a Top-Level menu bar. For example: \a&Help generates a Help command that is at the far right side of the menu bar.
-	The hyphen character causes a horizontal separator to appear in a menu.

CharacterCurrency Property

This property controls the character used to indicate a particular unit of currency. The default is the "$" symbol.

CharacterDecimal Property

This property controls the character used to denote a decimal point. The default is the "." symbol.

CharacterFalse Property

This property controls the single character a user must enter to indicate a false or negative response. The default is the "F" character. This property is used in conjunction with the CharacterTrue property.

CharacterThousands Property

This property controls the character used to separate every group of three digits to the left of the decimal point. The default is the "," character. Example: 1,000,000.00.

CharacterTrue Property

This property controls the single character a user must enter to indicate a true or positive response. The default is the "T" character. This property is used in conjunction with the CharacterFalse property.

Checked Property

This property, set to True or False, lets you control whether the object is checked by default. At runtime, your application can use the Checked property to learn whether a user turned a Menu Item or Check Box on or off. An ObjGet icon can retrieve the current setting of the object's Checked property. Your application can branch accordingly.

Note: Because Radio Buttons act as a group, you must use a different property to learn whether a Radio Button is On or Off. For a Radio Button, you must use the CheckedRadioButton property

CheckedRadioButton Property

This property identifies the ObjectName of the specific Radio Button (in a group of Radio Buttons) that has been selected by the user. Remember, unlike Check Boxes, the user cannot select more than one button in a group of Radio Buttons. In order to be recognized as belonging to the same group, the Radio Buttons must all have the same FamilyName setting. This property is a runtime, "get-only" property.

ClipChildren Property

When this property is set to True, the child window is clipped. You may need to set this property to True if live objects appear to flash when information is displayed in the Window object. In order to set ClipChildren to True, you must include a Window icon and an ObjSet icon in your structure. The Window icon creates the window and the ObjSet icon sets the property.

In most situations IconAuthor recognizes when it has to clip child windows. There are however, some situations where child windows are not automatically clipped. In these cases, you need to explicitly set the ClipChildren property to True. (The property is False by default.)

IconAuthor does not automatically recognize and clip child windows when third party program information is displayed in the window. For example, if your application displays a Button object and then displays a Gold Disk animation file on the background, the Button flashes. The Button flashes because it is not automatically clipped and it is re-drawing itself to remain in view on top of the animation. In this same scenario, if you set ClipChildren to True, the Button does not flash because it is clipped.

Important: Do not use live Graphic objects that have a transparent color (set via the ColorTransparent property) if ClipChildren is set to True. The entire object, including the area designated as transparent, will be clipped. Incorrect information will be displayed in the transparent area.

ClipSiblings Property

ClipSiblings works similarly to ClipChildren. When this property is set to True, the object is clipped. You may need to set this property to True if live objects appear to flash when information is displayed. In order to set ClipSiblings to True, you must include an ObjSet icon in your structure.

ColorBackground Property

This property controls the background color of the object. The drop-down list for this property lets you use the Color Editor to select the color you want to use.

ColorFill Property

This property, available at runtime only, controls the color of the background of all text in a Text object. The drop-down list for this property lets you use the Color Editor to select the color you want to use.

ColorFrame Property

This property, available at runtime only, controls the color used for the border frame of a Text object. The drop-down list for this property lets you use the Color Editor to select the color you want to use.

ColorHighlight Property

This property, available at runtime only, controls the color of the border highlight of a Text object. The drop-down list for this property lets you use the Color Editor to select the color you want to use.

ColorShadow Property

This property, available at runtime only, controls the color of the border shadow of a Text object. The drop-down list for this property lets you use the Color Editor to select the color you want to use.

ColorSpacer Property

This property controls the color of the space between the editable box and the drop-down arrow. If you want to keep color consistency, set this property to the same color as the ColorBackground property. The drop-down list box lets you use the Color Editor to select a color or create a custom color.

ColorText Property

This property controls the color of all text in the object. The drop-down list box lets you use the Solid Colors dialog box to select a color or create a custom color.

ColorTransparent Property

This property lets you specify a color, used in the graphic, that you want to appear transparent at runtime. Any transparent part of the graphic is no longer considered an active part of the object. The drop-down list for this property lets you use the Color Editor to select the color you want to use.

Example: You can create a small, square bitmap that looks like a round, grey button on a black background. By setting the Graphic object's ColorTransparent property to black, the user only sees the round button at runtime.

Command Property

This property, available at runtime only, resets the command for an object. Use these commands to manipulate the following objects through IconAuthor's ObjSet and ObjGet icons:

Audio
Database
Graphic
IconAnimate
Movie
Variable

Audio and Movie Object Commands

Close	Closes the object and any files associated with it.
Cue	Optional command to prepare the object for play. Potentially reduces delay prior to the Play command.
Open	Prepares the object for play.
Pause	Pauses play.
Play	Plays the object.
Resume	Resumes play of a paused object.
Seek	Seeks to the location in a file (or on a CD) that was previously specified via the PositionSeek property.
Stop	Stops the object from playing.

Database Object Commands

DataSourceConfig	Allows dynamic data source registration. Uses the SQLText property for the text string that represents the data source information.
DataSourceConnect	Connects to a data source using the ConnectString property.
DataSourceDisconnect	Disconnects from the data source.
DataSourceExecuteSQL	Directly executes an SQL statement without creating a recordset. Uses the SQL string set in the SQLText Property
RecordsetClose	Closes the recordset and frees resources
RecordsetOpen	Opens a recordset by performing the query based on the SQL string set in the SQLText Property
RecordsetRefresh	Refreshes any non-grid bound controls.
RecordsetRequery	Runs the recordsets query again to refresh the records.
RecordDelete	Deletes the current record from the recordset.
RecordSeekFirst	Positions the current record on the first record in the recordset.
RecordSeekLast	Positions the current record on the last record in the recordset.
RecordSeekNext	Positions the current record on the next record in the recordset.
RecordSeekPrev	Positions the current record on the previous record in the recordset.
RecordSeekTo	Seeks to the record in the recordset that was previously specified via the PositionSeek property.

Graphic Object Commands

DrawToBackground	Draws the graphic to the background bitmap. For live graphic objects, this allows you to display the graphic with an effect, using the Effect property.

IconAnimate Object Commands

Close	Closes the object and any files associated with it.

Open	Prepares the object for play.
Pause	Pauses play.
Play	Plays the object.
Resume	Resumes play of a paused object.
Stop	Stops the object from playing.

Variable Object Commands

ClearAll	Clears the variable data within the Variable object.
ClearArray	Clears the specified variable array within the Variable object.
ClearSingle	Clears the specified variable within the Variable object.
GetAll	Stores the information from the Variable object into the IconAuthor variable table.
GetArray	Stores the specified variable array from the Variable object into the IconAuthor variable table.
GetSingle	Stores the specified variable from the Variable object into the IconAuthor variable table.
PutAll	Stores the variable data from the Variable object into the IconAuthor variable table.
PutArray	Appends or updates the specified variable set in the VariableName property from the IconAuthor variable table and loads it into the Variable object.
PutSingle	Appends or updates the specified variable array set in the VariableName property from the IconAuthor variable table and loads the elements of that array into the Variable object.

CommandOnCreation Property

This property sets the command for an object to execute as soon as it is created at runtime. The available settings are Cue, None, Open, and Play. Note that the Audio and Movie objects use MCI to play.

Audio objects:

Open	Prepares the object for play.
Play	Plays the object.

Database objects:

DataSourceConnect	Connects to a data source using the ConnectString property.
RecordsetOpen	Opens a recordset by performing the query based on the SQL string set in the SQLText Property

IconAnimate objects:I

Open	Prepares the object for play.
Play	Plays the object.

Variable objects:

GetAll	Stores the information from the Variable object into the IconAuthor variable table.

GetArray	Stores the specified variable array from the Variable object into the IconAuthor variable table.
GetSingle	Stores the specified variable from the Variable object into the IconAuthor variable table.
PutAll	Stores the variable data from the Variable object into the IconAuthor variable table.
PutArray	Appends or updates the specified variable set in the VariableName property from the IconAuthor variable table and loads it into the Variable object.
PutSingle	Appends or updates the specified variable array set in the VariableName property from the IconAuthor variable table and loads the elements of that array into the Variable object.

ConnectExclusive Property

This property, set to True or False, makes an exclusive data source connection for the DataSourceConnect command. If True, the data source can only be used by that Database object.

ConnectString Property

This property lets you choose a data source from a drop-down list. To connect to a data source, the ConnectionString property must be filled in using the following format:

Datasource<;UserID><;Password><;Options>

Datasource represents the name of an installed ODBC data source. At edit time the connect string field will contain a list of all installed data sources in the drop-down list. Only those database files that you have added as data sources via the ODBC Administrator will appear in this list.

UserID is an optional user id.

Password is an optional password.

Options is any other connection information required by the data source.

See the section on Database Objects in Chapter 9 for information on adding data sources.

ControlBar Property

This property, set to True or False, sets whether a control bar appears below a Movie object. The control bar lets the user control how the video or animation plays. These controls also let you, the author, preview the movie within the SmartObject Editor.

ControlCommand Property

This property lets you set the Command property for Audio, Database, Graphic, IconAnimate, Movie and Variable objects. This property works in conjunction with the ControlObjectName property. The ControlObjectName property needs to be set to the ObjectName of the object you want to control.

ControlObjectName Property

This property lets you set the name of the SmartObject you want to control with the Button object. This property works in conjunction with the ControlCommand property. Once you have set the ControlObjectName property, you can set the ControlCommand property to the appropriate command.

CursorName Property

This property controls the way the cursor appears at runtime when it is over the object. The default setting is the arrow-shaped cursor. The drop-down list box lets you select one of the available cursors such as Indexed Hand, I Beam, and Crosshair.

CursorNameHotword Property

This property, set to a type of cursor, controls how the cursor appears over hotword in the Text object.

CursorPositionProgram Property

This property lets you get and set an X,Y coordinate of the cursor location on the active window.

CursorPositionScreen Property

This property lets you get and set an X,Y coordinate of the cursor location on the screen.

DataChanged Property

This property, set to True or False, lets you check to see if any updates have been made to the database file. Use an ObjGet icon on the DataChanged property and assign it a variable name. You can then choose to display any way you want.

DataFieldName Property

To bind an object to the Database object, set this property to the name of the field you want the object to display. The object will then be bound to that field. Select the Field Browser from the drop-down list to access the field names.

DataObjectName Property

To bind an object to the Database object, set this property to the same name as the Database objects ObjectName.

DataSources Property

This property displays a delimited list of all installed data sources. This is particularly useful if you want the user to be able to choose a data source. You could use an ObjGet icon to get the DataSources property and give it a variable name such as @datasources. You could then use an ObjSet icon to set the ItemList property to display the data sources list in a list box. The user would then be able to select a data source from the list box.

DataValueChecked Property

The database value you enter here will determine when the button will be checked or selected. You must enter the value exactly as it appears in your database file. Setting this property to a value that exists in a database field tells IconAuthor to compare the field contents to the contents of this property. If they match, IconAuthor will check or select the button.

DataValueUnChecked Property

The database value you enter here will cause the button to become unchecked when that value is displayed. You must enter the value exactly as it appears in the database record. Setting this property to a value that exists in a database field tells IconAuthor to compare the field contents to the contents of this property. If they match, IconAuthor will uncheck button.

DecimalPlaces Property

This property controls the number of digits that can appear to the right of the decimal point. The default is 0.

DefaultAction Property

This property controls the action that occurs when an OLE object executes (for example, when a user double-clicks on it). By default, this property is set to Server Default. Every server application has its own default behavior, for example, you *play* sound data and you *edit* Microsoft Word documents.

Optionally, you can set the DefaultAction to a value other than the server default. For example, a Sound object that contains MIDI data actually has three available actions: play, edit, and none. Setting DefaultAction to play plays the sound data when the object executes. If you set it to edit, the Microsoft Windows Sound Recorder appears, enabling the user to edit the data. If you set the property to none, the object does nothing.

An OLE object has different actions depending on the type of data it contains. After you insert data into an object, open the Properties dialog box, and use the drop-down list for the DefaultAction property to view the available actions.

DeleteProtected Property

This property, set to True or False, indicates whether (at runtime) the object can be deleted by an ObjDelete icon. An ObjDelete icon has a text box called Scope that lets you specify which objects to delete: one object, a class of objects, a family of objects, or All objects.

Dragable Property

This property, set to True or False, controls whether the object can be dragged by the user. The default is False.

DragAction Property

This property, set to Click or Drag, defines the action required by the end user to grab an object so that it can be moved. The value Click lets the user click the left mouse button on a dragable object, move the cursor to move the object, and click the left mouse button to drop the object. The value Drag lets the user press and hold the left mouse button on the object, drag the object to a new position, and release the left mouse button to drop the object. The default is Drag.

DragBringToTop Property

This property, set to True or False, controls how the object appears in relation to other live objects on the screen. Whenever a user clicks or drags an object to move it, the object automatically comes to the top screen layer. After the object has been dropped (or the move has been aborted), if this property is True, the object remains on top of all other live objects. If this property is set to False, when the object is dropped (or the move has been aborted) it returns to its original layering position. This means that the object may be partially or entirely obscured if other live objects are closer to the top screen layer and are located in the same region.

DragCursor Property

This property, set to True or False, determines whether a cursor is visible on top of the object as it is dragged. If you specify True, the default cursor for the object you are dragging appears (this is set via the CursorName property). If you specify False, no cursor appears.

DragGraphicNo Property

This property lets you specify the form you want a dragged object to take when it is *not* positioned over a valid drop target. You can set this property to a graphic filename. The drop-down list box lets you use the Browser to find the name of the file you want to use. This file must have the same dimensions as the file you specify for the DragGraphicYes property. If you do not specify a filename, the object's primary graphic (assigned to the FileName property) is used by default.

DragGraphicYes Property

This property lets you specify the form you want a dragged object to take when it is positioned over a valid drop target. You can set this property to a graphic filename. The drop-down list box lets you use the Browser to find the name of the file you want to use. This file must have the same dimensions as the file you specify for the DragGraphicNo property. If you do not specify a filename, the object's primary graphic (assigned to the FileName property) is used by default.

DragMode Property

This property, set to Move or Copy, controls what happens to the appearance of the original object when a user takes action on a dragable object. Move causes the original object to be moved to the new location as the user drags. Copy causes the original object to remain in its location; a copy of the object is dragged.

Note: A copy can be dragged away from the original object but a new object is not created. When the user drops the copy of the object, it disappears.

DragReturnOnFail Property

This property, set to True or False, determines whether the object returns to its previous location if it is dropped in an invalid position. If set to True, the object snaps back to its original location if the user drops it in an invalid position. If set to False, the object remains in the new position when it is dropped.

DragTargetName Property

This property is get-only. It is automatically set to the ObjectName of the target, when the DropType property of the target and the DragType property of the dragged object match, or when the DropType property is ALL. After a successful drop, your structure can use an ObjGet icon to retrieve the current setting of the DragTargetName property to learn exactly where the drop occurred.

DragTransparentColor Property

Use this property to make a color in the DragGraphicNo and/or DragGraphicYes graphics transparent. Set this property to a color or none. This allows the dragged graphics to have a transparent background. The drop-down list box lets you use the Solid Colors dialog box to select a color.

DragType Property

This property lets you identify where a dragable object can be dropped. A dragable object has a DragType value and a target object has a DropType value. Only if one object's DragType value matches another object's DropType value can the dragable object be dropped there.

Set the DragType property to a string (such as red), a semi-colon delimited list of strings (such as apple;pear;plum), or the keyword all. A single text string such as red, means the object can only dropped on a target object with a DropType that includes the text string red (for example red or the list red;blue;green). A list of strings means that the object can be dropped on a target object with a DropType that includes any of the items in the list. If an object has a DragType set to all, it can be dropped on any live Graphic object.

DrawStyle Property

This property determines how information appears in the object. Possible values for the property vary depending on the object class.

Related Topics:

DrawStyle Property and Graphic Objects
DrawStyle Property and OLE Objects

DrawStyle Property and Graphic Objects

The drop-down list options are: Scale, Clip, Tile, and Size By Graphic.

Scale	The graphic is resized to fit precisely into the available object. If the size of the object is not in proportion to the original graphic, the graphic may be stretched or compressed either horizontally or vertically.
Clip	The object contains as much of the graphic as is possible. If the object is smaller than the graphic some of the graphic will be hidden from view. If the object is larger than the graphic some white space will be visible where the graphic doesn't fill the object.
Tile	The graphic is taken in its original size, and is repeated as many times as necessary to fill the area of the object.
Size By Graphic	The object size automatically changes so that it precisely surrounds the entire graphic.

DrawStyle Property and OLE Objects

For an OLE object, this property determines whether the server or the SmartObject Editor controls the size of an OLE object. You can set this property to Size By Server or Size By Object.

Size By Server	The data you embed in the object controls the size of the object. Example: A chart created with Microsoft Graph will be the same size as it is in the server application.
Size By Object	The size of the object controls how the data appears regardless of how the data appeared in the server.

DropPosition Property

This property, set to centered or none, defines how a dragged object is positioned on the target object when it is dropped. If you specify centered, a dropped object is automatically centered on the target. If you specify none, a dropped object lands wherever it is positioned by the user.

DropType Property

This property lets you identify where a dragable object can be dropped. A dragable object has a DragType value and a target object has a DropType value. Only if one object's DragType value at least partially matches another object's DropType value can the dragable object be dropped there.

Set the DropType property to a string (such as red), a semi-colon delimited list of strings (such as apple;pear;plum), or the keyword all. A single text string such as red, means that only objects whose DragType includes the text string red (for example red or the list red;blue;green) can be dropped on the target. A list of strings such as apple;pear;plum means that only objects whose DragType includes one of these three text strings can be dropped on the target. If an object has a DropType set to all, any dragable object can be dropped on it.

Editable Property

This property, set to True or False, controls whether a user can edit the text in a Text object. The default is False.

Effect Property

This property lets you choose the effect to be used when the object displays. Static Graphic objects can de drawn into the background with an effect. Live Graphic objects can have their graphic drawn to the background with an effect (via the DrawToBackground setting of the Command property) but the objects Visible property must first be set to False. After the bitmap is drawn to the screen using the effect, set the Visible property to True to enable standard live object functionality. Choose Effect Selector... from the drop-down list to access the Effect Selector dialog box.

EmbeddedType Property

When you embed a graphic file, this property lets you specify what format the graphic information will be saved in within the SmartObject file. IconAuthor supports the following embedded graphic formats:

.BMP
.GIF
.RLE
.JPEG
.PCT

Enabled Property

This property, set to True or False, determines whether an object is enabled. When an object is enabled a user can interact with it. For example, if you disable a Push Button a user cannot click on it. Similarly, if you disable a List Box a user cannot scroll it or make a selection from it. When you disable an object any text labels it contains are greyed.

EnableFirst Property

This property, set to True or False, controls whether the Control Bar button that returns the first record when clicked is enabled.

EnableLast Property

This property, set to True or False, controls whether the Control Bar button that returns the last record when clicked is enabled.

EnableNext Property

This property, set to True or False, controls whether the Control Bar button that returns the next record when clicked is enabled.

EnablePrev Property

This property, set to True or False, controls whether the Control Bar button that returns the previous record when clicked is enabled.

EnableUpdate Property

This property, set to True or False, controls whether the Update Control Bar button is enabled.

FamilyName Property

This property allows you to optionally specify a family (group) to which an object belongs. If you make an object part of a family, at runtime, you can use an ObjSet icon to change a property of multiple objects simultaneously by specifying the change to effect a scope of family. You can also use an ObjDelete icon to delete a group of objects that belong to the same family.

Note: The FamilyName property provides special functionality for Radio Buttons (Button objects with Radio Button style). When you want a quantity of Radio Buttons to act as one group (where only one can be selected at any time), make sure to give each button the same FamilyName.

FieldCount Property

This property returns a count of the fields in the recordset. You can use an ObjGet icon on the FieldCount property and give it a variable name such as @fieldcount. Your application can evaluate the variable and branch accordingly.

FieldNames Property

This property returns a list of the field names in the recordset. You can use an ObjGet icon on the FieldNames property and assign it a variable name.

FileName Property

This property specifies the name of the file used by the object. The drop-down list box lets you use the Browser to find the name of the file you want to use. Files are automatically linked to Audio, Movie, Button and IconAnimate objects. Linked files remain separate from the SmartObject file. You have a choice of linking or embedding files for Graphic and Text objects. Embedded files become part of the SmartObject file. After you select a file for a Graphic or Text object, a File Access dialog box appears. Click on Link or Embed and choose OK to close the dialog box.

FileNameDisabled Property

This property lets you enter the filename containing the disabled graphic image for the button.

FitToWindow Property

This property, set to True or False, controls how information is displayed within a window. If True, information is scaled to fit in the available window. If False, information is not scaled.

Focus Property

This property, available at runtime only, allows you to set the focus to a particular object. When an object has focus it is the object that responds to any valid keyboard actions. For example, if a button has focus and the user presses the RETURN key, that button will be activated. This means that if one object has focus and you give focus to a second object, the first object no longer has focus.

Font Property

This property, available at runtime only, specifies the font characteristics of text. The drop-down list lets you use the Font dialog box to select the font characteristics you want to use.

FttPageName Property

Use this property to specify the page to display in the Text object. This property can only be set when a .ftt file is specified in the FileName property. Choose the Page Selector from the drop-down list for a list of pagenames in the .ftt file.

HasMouse Property

This property, set to True or False, indicates if the system IconAuthor or Present is running on has a mouse. You can use an ObjGet icon on the HasPen property and give it a variable name. Your application can evaluate the variable and branch accordingly.

HasPen Property

This property, set to True or False, indicates if the system IconAuthor or Present is running on has a pen. You can use an ObjGet icon on the HasPen property and give it a variable name. Your application can evaluate the variable and branch accordingly.

Height Property

Resets the height of the object in pixels. Specify a whole number greater than 0. This property is available at runtime only.

Hotword Property

The SmartObject Editor lets you can designate a character, word, or phrase in a Text object as a Hotword. This get-only property lets your application detect which Hotword a user clicked on in a Text object. If NotifyOnClickHotword is True, when a user clicks on a Hotword at runtime, a "ClickHotword" event is generated. Your structure can include an ObjEvent icon to await a "ClickHotword" event. Once the event occurs, an ObjGet icon can retrieve the current setting of the Hotword property and store it in a variable. Your application can evaluate the user's Hotword selection and branch accordingly.

HotWordActivate Property

This property is available for Text objects via the ObjSet icon. It works in conjunction with the HotWordColor Property. When you set the HotWordActivate property to a hotword index, all hotwords assigned that hotword index in the specified object(s) are automatically set to the color specified in the HotWordColor property.

HotWordColor Property

This property, set to a solid color, causes a hotword in the object to change to the specified color when the user clicks on it. The drop-down list box lets you access the Solid Color Editor to choose a color value from a drop-down list.

HotWordHighlight Property

Set this property to True to cause a hotword within the object to be reverse highlighted when the cursor is over it.

HotwordIndex Property

This get-only property lets your application detect the index of the Hotword a user clicked on. Within the SmartObject Editor you can assign a numerical index to a Hotword and the same index can be shared by other Hotwords. This is particularly useful in cases where you want the same action to occur for multiple Hotwords. For example, the Hotwords "car" and "bus" can both have an index of 2 and the Hotwords "canoe" and "rowboat" can have an index of 3. When your application detects that a "ClickHotword" event has occurred, it retrieves the current setting of the HotwordIndex property and stores it in a variable. Your application can evaluate the index number and branch accordingly.

IconFilename Property

This property controls the file used for an Icon style button. The drop-down list lets you use the Browser option to select the .ICO file you want to use.

Information Property

This get-only property lets your application detect what kind of device is associated with an object. For example, you can use an ObjGet on an Audio object and store the name of the device (e.g. MIDI Sequencer) in a variable. The Audio and Movie objects use MCI to play. The Information property represents the MCI Info command.

InputLimit Property

This property specifies the maximum number of characters a user can enter via the keyboard.

InputLimitBeep Property

This property, set to True or False, controls whether a system beep occurs if a user attempts to exceed the maximum number of characters allowable in an editable Text object. (The maximum number is set via the InputLimit property.)

InputTerminationRequired Property

This property, set to True or False, controls whether a user is required to press RETURN (when typing input) in order to generate a "Complete" event. Be aware that the NotifyOnComplete property must be set to True for the object in order for "Complete" event to occur. Given that NotifyOnComplete is True, when InputTerminationRequired is set to True, the user must press Enter. In the same situation when InputTerminationRequired is False, a "Complete" event is generated as soon as the user types the number of characters specified by the InputLimit property.

ItemList Property

This property sets the items that appear in a list box (open or drop-down). Separate one item from another with a semicolon (";"). For example, aqua;blue;green;purple will set the list to those items, in the specified order. If you want to have a semicolon appear as an actual part of a list item, precede it with a "\". However, remember to include the semicolons that are required as separators. As an example, blue\;;red\;;green\; appears as follows in the List Box at runtime.

Note: Within the SmartObject Editor you can also right click on an object and choose Item List... to enter list items via a dialog box.

KeyboardForward Property

This property, set to True or False, controls whether the keyboard information (the name of the key that was pressed) is "forwarded" to any other object as well. If KeyboardForward is False (this is the default) the other object does not get the keyboard information. If KeyboardForward is True, the other object does get the information.

KeyboardKeyPressed Property

When a user's action generates a "KeyDown" or "KeyUp" event, this property is reset to the name of the key. Once your application detects that a "KeyDown" or "KeyUp" event has occurred, an ObjGet icon can retrieve the current setting of the KeyboardKeyPressed property and store it in a variable. The application can then take action based on the user's selection.

KeyboardList Property

Set this property to specify the keys that the user can press. Use a semi-colon to separate one key or key combination from another.

KeyboardTabStop Property

This property, set to True or False, indicates whether a user can use the TAB key to set focus on this object at runtime.

Label Property

This property lets you specify the text that labels a button. The default is Button.

LabelType Property

This property lets you specify either Text or Graphic for the button label. If you choose Text, use the Label property to set the text. If you choose Graphic, two additional properties will appear in the properties list: FileName and FileNameDisabled. Enter the name of the graphic file in the FileName property. Enter the name of the disabled graphic file in the FileNameDisabled property.

Layer Property

This property lets you specify the layer hierarchy of an object. The Window object is always 0. Each object above the Window object is 1 greater than the previous object. Except for the System object, all live objects, including invisible live objects, have a layer value. You can set the Layer property at runtime with the following values:

A number greater than 0	Sets the object layer to that value.
+(number), example +3	Increases the layer by 3.
-(number), example -3	Decreases the layer by 3.
Bottom	Sets the layer to 1.
Top	Sets the layer so that the object is on top.

Left Property

Resets the distance (in pixels) from the left side of the page to the left side of the object. Specify a whole number greater than 0. This property is available at runtime only.

Length Property

This get-only property lets your application detect the length of a file (in milliseconds) or a CD (in Tracks:Minutes:Seconds:Frames). For example, you can use an ObjGet on an Audio object and store the length of the specified wave audio file in a variable. The Audio and Movie objects use MCI to play. The Length property represents the MCI Status command with the Length argument.

LightSource Property

This property, available at runtime only, controls the imaginary light source that lends shadows to a Text object. Set LightSource to one of four directional values: NW, NE, SE, or SW.

LineSpace Property

This property, available at runtime only, controls the line spacing in a Text object. Set LineSpace to one of three values: 1 (for single spacing), 1.5 (for one and one-half spacing), or 2 (for double spacing).

Location Property

Resets the pixel coordinates of the upper left corner of the object. Specify two numbers separated by a comma, for example: 250,125. This property is available at runtime only.

Mask Property

Set this property to a series of special characters that strictly controls which characters a user can enter and how many. Use the following reserved characters to create a mask:

9 - a number

A- any alphabetical character or space

X - any alphabetical character, number, or space

DD - a two-digit sequence that represents a day (a value from 01-31)

MM - a two-digit sequence that represents a month (a value from 01-12)

YY - a two-digit sequence that represents a year (01-99)

YYYY - a four-digit sequence that represents a year (0001-9999)

B - any character or symbol

! - any character or symbol where any alphabetical character is converted to uppercase

You can also use any character other than the above reserved characters as part of the mask. These characters will not be editable. Example: in the mask (999)999-9999, the parentheses and hyphen will not be editable. The characters the user types will appear only in the positions where there are 9s.

Note: If you need to use one of the reserved characters as part of a mask, precede the character with a \ symbol.

Maximized Property

This property, set to True or False, controls whether a Window object is maximized (full screen).

MenuItemCount Property

This property returns the total number of menu items in a menu.

MenuItemList Property

This property returns a semicolon separated list of the Menu Item object names in a Menu object.

Minimized Property

This property, set to True or False, controls whether a Window object is minimized to an icon.

MouseButton Property

This property, set to Left or Right, controls which mouse button the user employs to select a menu item from a pop-up menu. If you set it to Left, the user must left click to select an item from the menu. If you set it to Right, the user can either right click or left click. (This property is available for Windows 3.1 only.)

MultiMediaDeviceNames Property

This property displays a list of the multimedia product device names that are supported on the system that is running IconAuthor or Present. Use an ObjGet icon on the MultiMediaDevices property and assign it to a variable. Your application can then evaluate the contents of the variable and branch accordingly.

MultiMediaDevices Property

This property returns a list of the multimedia devices that are supported on the system that is running IconAuthor or Present.

MultipleLines Property

This property, set to True or False within the SmartObject Editor, controls whether an object can contain multiple lines of text. The default (True) causes text to wrap down vertically onto the next line when the cursor reaches the right side of the object. If this property is set to False, text does not wrap to the next line. Rather, it scrolls horizontally out of view when the Text object becomes full. The user can press the arrow keys to bring text back into view.

NotifyOnClickHotword Property

Use the NotifyOnClickHotword property to detect when a user clicks on a Hotword. The SmartObject Editor lets you designate a character, word, or phrase in a Text object as a Hotword. You can also assign a numerical index to a Hotword and the same index can be shared by other Hotwords. This is particularly useful in cases where you want the same action to occur for multiple Hotwords. If NotifyOnClickHotword is True, when a user clicks on a Hotword at runtime, a "ClickHotword" event is generated. Your structure can include an ObjEvent icon to await the event. Once the event occurs, an ObjGet icon can retrieve the current setting of the Hotword property or the HotwordIndex property and store it in a variable. Your application can evaluate the user's selection and branch accordingly.

NotifyOnClickLeft Property

This property, set to True or False, controls whether an event occurs when a user left clicks on an object. If the property is True and a user clicks left on the object, a "ClickLeft" event occurs. If it is False, no event occurs.

After your application displays an object upon which the user can click left, the structure typically contains an ObjEvent icon that waits to detect and record the result of the event. When an ObjEvent icon detects that an event has occurred it places the string "ClickLeft" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name.

NotifyOnClickMiddle Property

This property performs almost identically to the NotifyOnClickLeft property with two exceptions: 1) it applies to clicking the middle (rather than left) mouse button, and 2) it places the string "ClickMiddle" (rather than "ClickLeft") in the system variable @_Object_Event.

NotifyOnClickRight Property

This property performs almost identically to the NotifyOnClickLeft property with two exceptions: 1) it applies to clicking the right (rather than left) mouse button, and 2) it places the string "ClickRight" (rather than "ClickLeft") in the system variable @_Object_Event.

NotifyOnClose Property

This property, set to True or False, controls whether IconAuthor is alerted when the Window object is closed. When the Window is closed and this property is True, the string "Close" is assigned to the system variable @_Object_Event. Note that even if this property is set to false, a Window object may be closed; but IconAuthor is not alerted.

NotifyOnComplete Property

This property, set to True or False, controls whether a "Complete" event is generated by an object. When NotifyOnComplete is True, Audio, IconAnimate, Movie, and OLE Objects generate a "Complete" event when the object finishes playing. For a Text object, a "Complete" event is generated when a user presses the termination key while typing/editing text in an editable Text object. By default, the terminator key is Enter. When an ObjEvent icon detects that an event has occurred, it places the string "Complete" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name.

NotifyOnDoubleClick Property

This property performs almost identically to the NotifyOnClickLeft property with two exceptions: 1) it applies to double-clicking the left mouse button, and 2) it places the string "DoubleClick" (rather than "ClickLeft") in the system variable @_Object_Event. For more information see the NotifyOnClickLeft property.

NotifyOnDragAbort Property

This property set to True or False, controls whether an event occurs when a user clicks the right mouse button to abort the move or copy of a dragable object. When an ObjEvent icon detects that an event has occurred, it places the string "DragAbort" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name. Note that if this property is set to False, a user can abort a drag operation but an event does not occur.

NotifyOnDragDrop Property

This property set to True or False, controls whether an event occurs when a user drops a dragable object on a valid target. When an ObjEvent icon detects that an event has occurred, it places the string "DragDrop" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name. Note that if this property is set to False, a user may be able to drop an object in a valid position; but an event does not occur.

NotifyOnDragFail Property

This property set to True or False, controls whether an event occurs when a user drops a dragable object on an *invalid* target. When an ObjEvent icon detects that an event has occurred, it places the string "DragFail" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name. Note that if this property is set to False, a user may be able to drop an object in an invalid position; but an event does not occur.

NotifyOnEnter Property

This property, set to True or False, controls whether an "Enter" event occurs when a user moves the cursor over an object. This property is often used with the NotifyOnLeave property. (The NotifyOnLeave property generates a "Leave" event when the user moves the cursor off of an object.)

Example: You can set up your application so that when the user moves the cursor over a button, an "Enter" event occurs. When the application detects the event, a status bar at the bottom of the screen displays the purpose of the button. When the user moves the cursor off of the button, a "Leave" event occurs. When the application detects the "Leave" event, the status bar message disappears.

NotifyOnEnterHotWord Property

This property, set to True or False, controls whether an "Enter" event occurs when a user moves the cursor over an object. This property is often used with the NotifyOnLeave Property (The NotifyOnLeave property generates a "Leave" event when the user moves the cursor off of an object.)

NotifyOnError Property

This property, set to True or False, controls whether an "Error" event occurs when an Audio or Movie object generates an MCI error message. If this property is True and an error occurs, the Result property is set to the MCI error number and the ResultString property is set to the description of the MCI error.

NotifyOnGetFocus Property

This property, set to True or False, controls whether a GetFocus event occurs when the focus is set to the object. The user sets focus to the object by clicking on it or tabbing to it. This property is often used with the NotifyOnLoseFocus Property.

NotifyOnInput Property

This property, set to True or False, controls whether an event occurs when a user enters text in a Text object. When an ObjEvent icon detects that an event has occurred, it places the string "Input" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name.

NotifyOnInputLimit Property

This property, set to True or False, controls whether an event occurs when a user is entering text in a Text object and reaches the input limit. The input limit is set via the InputLimit property. When an ObjEvent icon detects that an event has occurred, it places the string "InputLimit" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name.

NotifyOnKeyDown Property

This property, set to True or False controls whether a "KeyDown" event is generated when the user presses a key that is defined in the KeyboardList property. Your application can include an ObjEvent icon to await the user's interaction. When the user interacts, the "KeyDown" event is stored in @_Object_Event and name of the Keyboard object is stored in @_Object_Name. A Branches composite can test these values to detect when a user activates a key. Use the KeyboardKeyPressed property to learn which key was activated.

NotifyOnKeyUp Property

This property, set to True or False controls whether a "KeyUp" event is generated when the user releases a key that is defined in the KeyboardList property. Your application can include an ObjEvent icon to await the user's interaction. When the user interacts, the "KeyUp" event is stored in @_Object_Event and name of the Keyboard object is stored in @_Object_Name. A Branches composite can test these values to detect when a user activates a key. Use the KeyboardKeyPressed property to learn which key was activated.

NotifyOnLeave Property

This property, set to True or False, controls whether a "Leave" event occurs when a user moves the cursor off of an object. This property is often used with the NotifyOnEnter property. (The NotifyOnEnter property generates an "Enter" event when the user moves the cursor onto an object.)

NotifyOnLeaveHotWord Property

Use this property to make hotwords within a Text object cursor-sensitive. This property, set to True or False, controls whether a "Leave" event occurs when a user moves the cursor off of a hotword. If NotifyOnLeaveHotWord is True for a Text object, an event called LeaveHotWord occurs when the user moves the cursor off of any hotword in the object. This property is often used with the NotifyOnEnterHotWord Property.

NotifyOnLoseFocus Property

This property, set to True or False, controls whether a LoseFocus event occurs when the focus is removed from the object. This property is often used with the NotifyOnGetFocus Property.

NotifyOnMaximize Property

This property, set to True or False, controls whether IconAuthor is alerted when the Window object is maximized. When the Window is maximized and this property is True, the string "Maximize" is assigned to the system variable @_Object_Event. Note that even if this property is set to false, a Window object may be maximized; but IconAuthor is not alerted.

NotifyOnMinimize Property

This property, set to True or False, controls whether IconAuthor is alerted when the Window object is minimized. When the Window is minimized and this property is True, the string "Minimize" is assigned to the system variable @_Object_Event. Note that even if this property is set to false, a Window object may be minimized; but IconAuthor is not alerted.

NotifyOnPressLeft Property

Use this property, set to True or False, to 1) allow users to interact with touch screens and 2) set up an object as press and hold sensitive. When the property is True and the user touches the object or presses the left mouse button on the object, the event "PressLeft" is generated. Your application can use an ObjEvent icon to await the user's interaction. When the user interacts, "PressLeft" is stored in @_Object_Event and the name of the affected object is stored in @_Object_Name.

NotifyOnPressRight Property

This property works similarly to the NotifyOnPressLeft property. Use this property, set to True or False, to set up an object as press and hold sensitive. When the property is True and the user touches the object or presses the right mouse button on the object, the event "PressRight" is generated. Your application can use an ObjEvent icon to await the user's interaction. When the user interacts, "PressRight" is stored in @_Object_Event and the name of the affected object is stored in @_Object_Name.

NotifyOnRecordFirst Property

This property, set to True or False, controls whether an event occurs when the user clicks on the First Control Bar button. The event RecordFirst is stored in the variable @_Object_Event.

NotifyOnRecordLast Property

This property, set to True or False, controls whether an event occurs when the user clicks on the Last Control Bar button. The event RecordLast is stored in the variable @_Object_Event.

NotifyOnRecordNext Property

This property, set to True or False, controls whether an event occurs when the user clicks on the Next Control Bar button. The event RecordNext is stored in the variable @_Object_Event.

NotifyOnRecordPrev Property

This property, set to True or False, controls whether an event occurs when the user clicks on the Previous Control Bar button. The event RecordPrev is stored in the variable @_Object_Event.

NotifyOnRecordUpdate Property

This property, set to True or False, controls whether an event occurs when the user clicks on the Update Control Bar button. The event RecordUpdate is stored in the variable @_Object_Event.

NotifyOnSelect Property

This property, set to True or False, controls whether an event occurs when a user makes a final selection from the object. If the property is True and the user does one of the following, a "Select" event occurs.

Combo Box	The user types in the text box and presses Return.
Combo Box	The user clicks on an item.
Combo Box	The user presses the arrow keys to change the highlighted list item and presses Return.
List Box	The user double-clicks on an item.
List Box	The user selects an item and presses the Return key.

When an ObjEvent icon detects that an event has occurred, it places the string "Select" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name. Note that if this property is set to False, a user may be able to make a selection from a list box; but an event does not occur.

NotifyOnSelectChange Property

This property, set to True or False, controls whether an event occurs when a user changes the currently highlighted item in the object. If the property is True and the user does one of the following, a "SelectChange" event occurs.

Combo Box	The user presses the arrow keys.
List Box	The user presses the arrow keys.
List Box	The user clicks on a different item.

When an ObjEvent icon detects that an event has occurred, it places the string "SelectChange" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name. Note that if this property is set to False, a user may be able to change the highlighted item, but an event does not occur.

NotifyOnSize Property

This property controls whether IconAuthor is alerted when the Window object is resized. The drop-down list lets you set NotifyOnSize to True or False. When the Window is resized and this property is true, the string "Size" is assigned to the system variable @_Object_Event. Note that even if this property is set to false, a Window object may be resized; but IconAuthor is not alerted.

NotifyOnStart Property

This property, set to True or False, controls whether a "Start" event is generated by an object. If NotifyOnStart is True an OLE Object generates a "Start" event when the object begins its action. When an ObjEvent icon detects that an event has occurred, it places the string "Start" in the system variable @_Object_Event. Also, the name of the object is placed in @_Object_Name.

ObjectData Property

Use this property to enter any information you want on the object. This property is only available for live objects and is only necessary if you plan to reference the object in IconAuthor icons.

At Runtime, when a user acts on an object (for example, clicks on a Button) the data of the object that was affected is automatically placed in the system variable @_Object_Data. This information is extremely valuable because your application can use If icons to evaluate the contents of @_Object_Data and branch accordingly.

ObjectName Property

This property is the unique name that you assign to an object within the SmartObject editor. This property is only available for live objects and is only necessary if you plan to reference the object in IconAuthor icons. For example, at runtime, if you want to change (set) or retrieve (get) the current property of an object, you will need to specify the object name in the ObjSet or ObjGet icon Content Editor.

Also at runtime, when a user acts on an object (for example, clicks on a Button) the name of the object that was affected is automatically placed in the system variable @_Object_Name. This information is extremely valuable because your application can use If icons to evaluate the contents of @_Object_Name and branch accordingly.

PageName Property

This property displays the name of the SmartObject page the object is on. The PageName property lets you group and manipulate objects at runtime in the same way as the FamilyName and ObjectName properties. You can enter the keyword pagename in the Scope field on any of the Obj- icons. This way you can set, get, or delete a property for all objects on a SmartObject page using one icon.

PauseOnMinimize Property

This property, set to True or False, causes your application to pause when the Window object is minimizes. When set to True, the application will pause. When set to False, the application will keep running.

PlayCount Property

This property set to a positive integer, controls how many times the specified file or CD selection plays. The default is 1.

PositionCurrent Property

This property, available at runtime only, is automatically set to the current position in a file or on a CD. Your application can use an ObjGet icon to retrieve the setting of this property and store it in a variable for subsequent display or manipulation. The format for this value is milliseconds for audio files, frames for movie and animation files, and Track:Minutes:Seconds:Frames for CD.

PositionEnd Property

This property sets the beginning position for playing a file or a CD selection. The format for this value is milliseconds for audio files, frames for movie and animation files, and Track:Minutes:Seconds:Frames for CD.

PositionSeek Property

This property sets the seek position for a file or a CD selection. Once you set the seek position you can set the Command property to Seek, thereby causing the object to find the specified position. The format for the PositionSeek value is milliseconds for audio files, frames for movie and animation files, and Track:Minutes:Seconds:Frames for CD.

PositionStart Property

This property sets the beginning position for playing a file or a CD selection. The format for this value is milliseconds for audio files, frames for movie and animation files, and Track:Minutes:Seconds:Frames for CD.

ProgramType Property

This property indicates the type of the system (i.e. UNIX, MAC, OS/2, Win16) that is running IconAuthor or Present.

ProgramVersion Property

This property indicates the version of the system that is running IconAuthor or Present.

RecordCount Property

This property displays the number of records in the recordset.

RecordData Property

This property lets you get and set data for the current record using the ObjGet and ObjSet icons. When using an ObjSet, it must be followed by a RecordAdd or RecordUpdate command.

RecordStatus Property

This property displays the status of the current record. The status can be: Success, Updates, Deleted or Error.

Rectangle Property

Resets the area of the object. This property is available at runtime only. Specify four numbers separated by commas to describe the screen coordinates of the object's upper left corner and the object's lower right corner.

Example: 50,50,100,100 specifies that the object's upper left corner is at 50,50 and the object's lower corner is at 100,100.

ResizeToFile Property

This property, set to True or False, controls the size of a Movie object. When the property is True (the default), the object automatically resizes itself to the appropriate dimensions for the specified file. When the property is False, the object maintains its original size (as drawn). Note that this may cause some of the image to be out of view if the object is not large enough.

Result Property

This property, is set when appropriate, to the number of an MCI error message.

ResultString Property

This property, is set when appropriate, to the description of an MCI error message.

ReturnToStart Property

This property, set to True or False, re-sets the frame number to the first frame when the movie is finished playing.

Right Property

Resets the distance (in pixels) from the left side of the page to the right side of the object. Specify a whole number greater than 0. This property is available at runtime only.

ScreenColors Property

This property returns the number of colors the system IconAuthor or Present is running on supports.

ScreenHeight Property

This property returns the screen height of the system that is running IconAuthor or Present.

ScreenPaletteEntries Property

This property returns the number of palette entries the system IconAuthor or Present is running on supports.

ScreenWidth Property

This property returns the screen width of the system that is running IconAuthor or Present.

ScrollBarVertical Property

Set to True or False, this property controls whether the object has a scroll bar.

SelectedItemData Property

This property is set to the item that the user selects (or possibly types in a Combo Box). An application can use an ObjGet icon to retrieve the current value assigned to this property and store it in a variable. This get-only property is available at runtime only.

SelectedItemNumber Property

This property is set to the number of the item that is currently selected. You can set this property to a pre-selected item number in the SmartObject Editor. Also, this property is set at runtime when the user makes a selection (or possibly types a value in a Combo Box). The first (topmost) item in a list is 1, the second item is 2, and so on. An application can use an ObjGet icon to retrieve the current value assigned to this property and store it in a variable.

SelectionArea Property

This property lets you assign a unique number to an object to identify it as a selection area (hotspot) on which the user can click the left mouse button at runtime. When you create pages that contain objects that are selection areas, you use the Menu composite which includes the InputMenu icon (versus object icons) to:

1. Display the page.
2. Allow the user to click on an area.
3. Store the area number in @_Selection.
4. Evaluate the contents of @_Selection and branch according to the user's choice.

ShowPartialItems Property

This property, set to True or False controls whether items in a list box scroll by partial items or by whole items. If the property is True, items appear gradually, similar to the numbers that gradually rotate into view on an odometer. If the property is False, each whole item pops into view in its entirety.

Size Property

Resets the width and height of the object (in pixels). Specify two numbers separated by a comma. For example: 200,56 means that the object is 200 pixels wide and 56 pixels tall. This property is available at runtime only.

Sort Property

This property, set to True or False within the SmartObject Editor, controls whether the items in a list box are sorted alphabetically.

SQLText Property

This property lets you enter a SQL string. The SQL string is a query that tells the database what information to search for and display. The SQL string is then executed by the RecordsetOpen command. A table of special keyterms has been provided for advanced users at the end of this section.

You can also use this property to register data sources on the fly. This is useful if you want to provide a complete installation for your customers. Enter the data source information in this property using the following format: ODBC driver description, list of driver specific attributes.

Each driver specific attribute should be followed by a backslash and the character zero. You can examine the ODBC.INI file to see what valid attributes are used by a particular driver. Also driver documentation may provide this information.

State Property

This runtime only property lets you set the state of an OLE object to Executing or Idle. If the state of the object is Executing the object's default action (set via the DefaultAction property) occurs. If the state is Idle, no action occurs.

Example: Your application displays a page with a non-visible OLE object that is defined to play a wave audio file. A user cannot double-click on the object to play it, however, an ObjSet icon can set the object's State property to Executing and the sound file plays.

Status Property

This property, available at runtime only, is automatically set to the current status of the object, for example, playing, opened, or closed. Your application can use an ObjGet icon to retrieve the current setting of the Status property and store it in a variable. As an example, if an audio object is playing and an ObjGet icon retrieves its status, the status will be expressed as "playing."

Style Property

This property, available at runtime only, changes the style of an object. A Button object can be set to Push Button, Radio Button, Check Box, Icon Button, or Group Box style. A Timer can be set to Periodic, Count Down, Count Up, or Alarm style.

TableCount Property

This property displays the number of tables in the data source. An ObjGet icon needs to get the TableNames property data first. Then you can use a second ObjGet icon on the TableCount property and store the number of tables to a variable.

TableNames Property

This property returns a list of the table names in the data source. Use an ObjGet icon on the TableNames property and assign the table names to a variable. This property will close the recordset in the Database object if it is open.

Text Property

This property, set to the text that the Text object currently contains, has two common uses. First, you can use an ObjSet icon to reset the Text property, thereby resetting the text that appears in the object. If you are typing characters into the ObjSet icon in order to set the Text property, the limit is 256 characters. If you are specifying a variable (that contains text characters) in the ObjSet icon, the limit is 2000 characters.

Second, you can use this property in an ObjGet icon to retrieve the value that a user typed in a Text object. For example, if a user types the word blue into an editable Text object, the Text property is set to blue. Your application can use an ObjGet icon to take the current Text property value and store it in a variable.

TextBoxData Property

This property lets you set the string of text you want to display in the edit field or get the string of text entered by the user.

TextCase Property

This property, set to MixedCase, UpperCase, or LowerCase, controls the case used for alphabetical characters. MixedCase leaves the text in the case in which it was entered. UpperCase converts all text to uppercase. LowerCase converts all text to lowercase.

TextDragableProperty

This property, set to True or False, controls whether the user can move the text around in an editable Text object.

TextFormatted Property

This property allows you to retrieve text from a Text object, including formatting such as carriage returns.

Example: A user types text (including carriage returns) into a small Text object. An ObjGet icon retrieves the TextFormatted setting for the input and stores it in @input. An ObjSet icon re-displays the text in another larger Text object (by setting the Text property to @input). Although the second Text object is larger, the text displays with the original formatting information.

TextLength Property

This property is set at runtime to the number of characters in a Text object. You can use an ObjGet icon to learn how many characters a user entered.

TimerData Property

This property lets you specify the data used by the Timer object. For a Periodic timer, this setting is the amount of time between alarm events. Example: 01:00:30. For a Count Up timer this is 0 (the starting point in time). For a Count Down timer this is the amount of time that passes before the alarm event occurs. Example: 00:00:10. For an Alarm timer this is the specific time of day at which the event should occur. Example: 14:00:00.

TitleBarText Property

This property lets you set the text that appears in the title bar of the window.

Top Property

Resets the distance (in pixels) from the top of the page to the top of the object. Specify a whole number greater than 0. This property is available at runtime only.

TrackCount Property

This property, available at runtime only, is automatically set to the number of tracks on the current CD. Your application can use an ObjGet icon to retrieve the setting of this property and store it in a variable for subsequent display or manipulation.

TrackLength Property

This property, available at runtime only, is automatically set to the length of the current track. Your application can use an ObjGet icon to retrieve the setting of this property and store it in a variable for subsequent display or manipulation. The format for this value is Track:Minutes:Seconds:Frames.

TrackNumber Property

This property, available at runtime only, is automatically set to the number of the current track on the current CD. Your application can use an ObjGet icon to retrieve the setting of this property and store it in a variable for subsequent display or manipulation. The format for this value is Track:Minutes:Seconds:Frames.

TransparentBackground Property

This property, set to True or False, controls whether the background color of the object is solid or transparent. This is particularly useful when you display data such as a Microsoft Paintbrush image.

ValidationList Property

This property returns a list of bound objects whose data has changed. The DataChanged property has to be set to True to be able to access the validation list.

VariableName Property

Enter the name of a variable you assigned using the Variable object. You can then manipulate the specified variable using the PutSingle command. Or, if you entered a variable array in the Variable object, enter the name you assigned the array. You can then manipulate the array using the PutArray command.

Visible Property

This property, set to True or False, indicates whether an object can be seen. When an object is not visible it still exists and can be made visible at some later time. Many object classes that have a Visible property, *except* for example OLE and Audio objects, are effectively disabled when you make them invisible. Example: If the user cannot see a Push Button or a List Box, he or she cannot click or double-click on it.

Keep in mind that it is faster to make an object visible than it is to delete the object and then re-display the entire page again. You may find it useful to create a page with more objects than you initially need in your display. Objects that are intended for later use can simply be set so they are not visible. As necessary, at runtime, an ObjSet icon can be used to change the property to visible.

Related Topics:

Visible Property and Transparent Objects

Visible Property and Transparent Objects

Even a Transparent object has a Visible property. Although even when visible, Transparent objects cannot be seen, this property can be useful because of its disabling capability.

VisibleFirst Property

This property, set to True or False, controls whether the First control bar button is displayed.

VisibleLast Property

This property, set to True or False, controls whether the Last control bar button is displayed.

VisibleLines Property

This property, set to True or False, controls whether the horizontal lines are visible.

VisibleNext Property

This property, set to True or False, controls whether the Next control bar button is displayed.

VisiblePrev Property

This property, set to True or False, controls whether the Previous control bar button is displayed.

VisibleUpdate Property

This property, set to True or False, controls whether the Update control bar button is displayed.

Width Property

Resets the width of the object in pixels. Specify a whole number greater than 0. This property is available at runtime only.

WidthEdge Property

This property, available at runtime only, controls the width of a Text object's border edge. Set WidthEdge to the desired number of pixels.

WidthFrame Property

This property, available at runtime only, controls the width of a Text object's border frame. Set WidthFrame to the desired number of pixels.

Variables

When you enter a **fixed** value (such as a number or a color) in a Content Editor field, every time you run the icon it performs in exactly the same manner. The alternative to using fixed values is to use **variables**. While a fixed value causes an icon to do the same thing every time, a variable can cause the icon to execute differently (vary) on different occasions.

A variable is similar to a container. Like a container that can hold oil on one occasion and water on another, a variable can contain one value at one time and hold a different value later. Unlike a physical container, a variable is an expression like @NAME, @NUMBER, or @COLOR that you enter in a Content Editor field. As an example, if you enter the variable @COLOR in the Clear Screen To field of the Clear icon, on one occasion, @COLOR can contain the value blue and later it can contain green.

Related Topics:

How Does a Value Get in a Variable
Types of Variables

Types of Variables

IconAuthor lets you use the following types of variables:

User Variables
Indexed Variables
System Variables
Path Variables

How Does a Value Get in a Variable?

When you use a variable in an application, the variable must contain a value to perform properly. There are several ways that a value gets stored in a variable.

You (the Author) Assign a Value

A common way to set a variable is to use a Variable icon. This icon lets you specify the name of the variable and the value you want it to contain. As an example, a Variable icon at the beginning of your application could set @COLOR=blue. As a consequence, anywhere in your application where the variable @COLOR appears, it will use the color blue.

It may seem trivial to set @COLOR to use it in one Clear icon. However, consider what would happen if you had a large application that used several Clear icons (and other icons) that used @COLOR. In order to change @COLOR globally you would only have to edit the value in one Variable icon.

The user can assign a value to a variable.

Applications often let the user enter information, such as a name or number, via the mouse or keyboard. In order to let different users enter different information, your application stores the input in a variable. For example, your application can contain a Display icon that displays a SmartObject file. One of the items that displays could be a Text object in which the user is instructed to type his or her name. After the user enters the name, the value can be stored in a variable called @NAME.

Using variables to capture a user's input and store it in a variable is extremely powerful. Once the value is stored in a variable, the variable can be used elsewhere in the application. The user can input many different kinds of information such as test responses, order information, or personal information.

An Icon Automatically Assigns a Value

A small number of icons actually place values into a variable for you. The Random icon is a good example of this practice. This icons Content Editor requires you to specify the name of a variable (such as @NUMBER) and a range of numbers (such as 1-10). At runtime, the icon randomly picks a number in the specified range and assigns it to the variable.

Some Basic Rules for User Variables

A variable can hold one or more numbers, sets of coordinates, text strings, or filenames.

All variables must begin with the @ symbol and can contain up to alphanumeric 19 characters, including the @ symbol. Do not begin a variable with @_. This convention is reserved for special system variables.

Valid characters for naming a variable are numbers, letters (upper or lowercase), and the underscore (_) symbol.

Variables can store numeric values from -2.1 billion to +2.1 billion.

Application variables can store text strings up to 256 characters in length.

Within one application, do not use the same name for a single variable and an indexed variable. For example, do not use @VAR and @VAR[1], @VAR[2], etc.

User Variables

Each variable you create must have a unique name. It is helpful to name variables in a way that indicates their use. For example if you create a variable that is intended to store a user's last name, you can call it @LAST_NAME.

The information stored in a variable can be a text string like "hello there" or "George", a number like 47, a coordinate point like 200,10, or a filename, like DOGS.PCX. The first character of every variable name you create must be the @ symbol. In addition to the @ symbol, use up to 18 upper and lower case letters, numbers, and the underscore symbol to name your variables. Variable names are not case sensitive. Do not begin a variable with "@_". This convention is reserved for special system variables.

Related Topics:

Capturing User Responses
Performing Concatenation
Performing Real Number or Integer Arithmetic Operations
Evaluating Input
Loading Multiple Variables

Capturing User Responses

Your application can let a user type information and store it in a variable. Once the data is in the variable, you can manipulate it in several different ways. For example, you can display a name, evaluate a test response, or save order information.

One of the most common ways to capture user input is to display a SmartObject file with an editable **Text object**. A Text object is a box designed to display text. If you make it editable, the user can type in it..(To make a Text object editable, you set its Editable property to True in the SmartObject Editor.)

When a user finishes typing in a field (an editable Text object), your application can store the input in a variable by using an ObjGet icon. In effect, the ObjGet icon gets the value currently associated with the objects Text property (this is simply the text that the user typed) and assigns it to a variable.

Performing Concatenation

You can link together the contents of two or more variables that contain character strings. This process is called **concatenation** and is done via the Variable icon. As an example, you can create an application that lets a user enter his or her name and re-displays the name as part of a personalized greeting.

Use the & operator to concatenate. For example, if @GREETING=Hello and @NAME=Fred, you can concatenate the two values in a Variable icon to construct the message Hello Fred. In the Variable Name field, specify the name of the variable (such as @MESSAGE) in which you want to store the result of the concatenation. In the Assign Contents field, specify @GREETING & @NAME. Be sure to include a space before and after any operators in the Variable icon.

In this example, as a result of the concatenation @MESSAGE will equal Hello Fred.

Perform Real Number or Integer Arithmetic Operations

If the information assigned to a variable is numeric you can perform arithmetic operations on it. You you can set up a Variable icon that adds, substracts, divides, multiplies, determines square roots, and so on. You can specify a simple operation such as @NUMBER1 + @NUMBER2 or you can specify more complex, multi-operator expressions, such as (@NUMBER1 + @NUMBER2) / @NUMBER3.

Remember to leave spaces before and after any operators. Parentheses ensure that an enclosed operation is done before other operations.

Evaluating Input

Once an application captures user input, it often evaluates the input and branches (executes one series of icons versus another) based on the evaluation. One of the most common ways to evaluate input is via the If icon.

Consider an application that lets a user enter a response to the question What is 50 + 25? The user enters a response which is stored in @ANSWER. You use an If icon to compare the users input to the number 75 (the correct answer). The If icon acts as a traffic controller by determining whether the response was correct, and directing execution flow in a particular direction.

Loading Multiple User Variables

Although the Variable icon is useful for assigning a value to a variable, if you are assigning values to several variables, it may be more efficient to load them into memory simultaneously. This can be done via the Display icon or by the LoadVar icon. Many types of applications can require you to load a group of variables at once.

As an example, consider an application that must display in one of several languages. Instead of fixed values, all of your icons that display information (such as bitmaps and SmartObject files) would use variables. For example, Display icons that show bitmaps would be set up to display @BITMAP1, @BITMAP2, and so on. At the beginning of the application a display can present the user with a selection of languages (such as English and Espanol). After the user makes a selection, the application can evaluate the result and load the appropriate group of variables and values. For example, one group assigns @BITMAP1 and @BITMAP2 filenames that display English information while another group assigns these variables using filenames that display Spanish information.

Related Topics:

The Display Icon and the Variable Object
The LoadVar Icon and Variable Files

The Display Icon and the Variable Object

To load several variables into memory at once, you can use the Display icon to load a SmartObject file with a Variable object. Use the Variable object to define one or more variables and their values. Although the Variable object has no visible appearance at runtime, the SmartObject file can optionally contain other objects that do appear, such as Text or Graphic objects.

As an example, a Variable object might load the following:

@NUMBER_OF_GRAPHICS
3
@COLOR
blue
@CORRECT_MESSAGE
Yes. That is correct.
@INCORRECT_MESSAGE
No. That is incorrect. Try again.

As soon as the SmartObject file is loaded into memory, the variables defined in the Variable object are also loaded and available for use. This method is efficient in two ways. First, it lets you use one Display icon instead of four Variable icons. Second, it lets you load variables that are associated with a particular SmartObject page.

The LoadVar Icon and Variable Files

The LoadVar icon also lets you load a group of variables into memory simultaneously. The variables you load with the LoadVar icon come from a **variable file** which is an ASCII text file. The file contains any number of variables and can be created with a text editor such as Notepad, or using a text editing window within IconAuthor. When you name a variable file, use a .VAR extension. Include a carriage return at the end of the last line in a variable file.

Elements listed in a variable file are not required to have values. The values can be assigned to the elements later in the execution of the application. For example, the application can capture user input to assign values to the elements.

To access Notepad from IconAuthor:

+ Choose Notepad from the Run menu.

Indexed Variables

Unlike user variables which can contain only one piece of information, **indexed variables** are user variables that can contain several pieces of related information. For example, an indexed variable might contain all the answers to a quiz or it might contain all the graphics to display as part of a presentation.

Each piece of information in an indexed variable is contained in its own **element**. For example, you can store the names of six graphic files that are part of one presentation in an indexed variable with 6 elements.

An element name has two parts to it: the variable name and the index. @PIC[1], @PIC[2], and @PIC[3] are examples of elements in one indexed variable. @PIC is the variable name, and [1], [2], and [3] are the indexes. @PIC[1] is an element that can contain the name of one graphic file, @PIC[2] can contain another, and so on. The index part of an indexed variable element can be a fixed value or a variable. For example, @PIC[1] uses a fixed value for the index. @PIC[@COUNT] uses a variable for an index.

Do not use the same variable name for a single variable and an indexed variable. For example, do not use @NUMBER and @NUMBER[1], @NUMBER[2], etc.

Note: Unless otherwise directed, always use the full name (variable name and index) to refer to an indexed variable. Do not, for example, refer to @PIC without an index.

Each element of an indexed variable can contain the same kind of information that a single value user variable can contain, such as a number or a character string, or another variable. The indexed variable elements can also be used in the same manner as single value application variables. That is, they can be used to capture user responses, perform arithmetic procedures, and so on.

Related Topics:

Indexed Variables and Loops
Indexed Variables and Parsing

Indexed Variables and Loops

In many situations, applications manipulate the elements of an indexed variable in sequence. Each element is used in order, according to the number of its corresponding index. If an indexed variable contains the elements @PIC[1], @PIC[2], and @PIC[3], an application is likely to be designed to manipulate @PIC[1] first, then @PIC[2], and then @PIC[3].

This use of indexed variables is illustrated by how the elements of the variable work within a **loop**. A loop is a group of icons in your structure that is executed more than once because execution flows around in a circular fashion. Rather than building the same logic into your structure repeatedly, a loop is efficient because it uses the *same* logic, or, the same portion of the structure over and over again.

Each time through a loop your application can use a different element of an indexed variable.

Indexed Variables and Parsing

In IconAuthor, **parsing** means breaking down a value in a variable into its smaller components. One of the most common uses of parsing is to store related pieces of information in one variable and at runtime, to parse the components into discrete, usable pieces of data. Use a Parse icon as the foundation for a parsing routine in your application.

System Variables

Unlike user variables, which you create as part of your applications, *system variables* are created and assigned values by IconAuthor. System variables are reserved for certain standard functions. All system variables begin with the characters @_. Although you can change the values stored in system variables, you cannot change the system variable names. Each system variable serves a unique purpose.

System Variables

@_ERROR	This variable is set when a DllLink icon executes. When the icon executes successfully this variable contains the value 0. When the icon executes and an error occurs, the variable contains a non-zero value.
@_ERROR_STRING	This variable is set when a DllLink icon executes. If an error occurs during execution this variable contains a message that describes the nature of the error.
@_FOUND	This variable is set when a Database icon is executed with a LOCATE, NEXT, or EXIST command. It is set to .T. if IconAuthor finds a valid database or database record. It is set to .F. if IconAuthor does not find a valid database or database record.
@_NUM_AREAS	This variable is set when a SmartObject file is displayed. It contains the number of selectable areas on the screen.
@_OBJECT_DATA	This variable is set when your application uses live objects. (Applications can use live objects that are created as part of SmartObject file.) When a live object event occurs (such as a button being clicked), the *setting of the objects ObjectData property* is stored in this variable.
@_OBJECT_EVENT	This variable is set when your application uses live objects. (Applications can use live objects that are created as part of SmartObject file.) When a live object event occurs (such as a button being clicked), the *name of the event* is stored in this variable.
@_OBJECT_NAME	This variable is set when your application uses live objects. When a live object event occurs (such as a button being clicked), the *name of the object* is stored in this variable.
@_RUN_WND	This variable is set to the window handle of the window in which your

	IconAuthor application is running. This variable can be passed to DLL functions that require a window handle.
@_SELECTION	This variable is set if a user makes a selection from an input screen generated by an InputMenu icon. It contains the number assigned to the chosen selection area. This variable is also used to store the key used to terminate input with the Input icon.
@_TEXT_AREAS	This variable is set when a SmartObject file is displayed. It contains the coordinates for the selectable areas on the screen.
@_TIMEOUT	This variable is set when an Input or InputMenu icon is executed. It is set to 1 if timeout occurs before the user responds to the input screen. It is set to 0 if the user responds before timeout.
@_USERTIME	This variable is set when an Input or InputMenu icon is executed. It is assigned the number of 100ths of a second it took the user to respond to the input screen.

Path Variables

Path variables are a specialized type of system variables because they are created and assigned values by IconAuthor. Like other system variables, they begin with the characters @_.

The way in which path variables differ is that they are always assigned values that are paths. For example, by default the path variable called @_GRAPHICS_PATH is assigned a value similar to the following: C:\IAUTHOR\GRAPHICS. The purpose of path variables is to tell an application where to find its files (such as graphics, animations, and other application files).

There is a different path variable for each type of file that your application uses.

@_ANIMATE_PATH	Contains the path for IconAnimate animation files, for example, C:\ IAUTHOR\ANIMATE.
@_AUDIO_PATH	Contains the path for audio files, for example, C\IAUTHOR\AUDIO.
@_DATA_PATH	Contains the path for database files, for example, C:\IAUTHOR\DATABASE.
@_FORMAT_PATH	Contains the path for database format files, for example, C:\IAUTHOR\ FORMAT.
@_GRAPHIC_PATH	Contains the path for graphic files, for example, C:\IAUTHOR\GRAPHICS.
@_ICONWARE_PATH	Contains the path for application files, for example, C:\IAUTHOR\ICONWARE.
@_INPUT_PATH	Contains the path for input template

	files, for example, C:\IAUTHOR\INPUT.
@_LIB_PATH	Contains the path for library files, for example, C:\IAUTHOR\LIBRARY.
@_MOVIE_PATH	Contains the path for digital video files and third-party animation files, for example, C:\IAUTHOR\TEXT.
@_TEXT_PATH	Contains the path for SmartObject files and ASCII text files intended for display, for example, C:\IAUTHOR\TEXT.
@_USER_PATH	Contains the path for user files, for example, C:\IAUTHOR\USER. User files are created and used only by Present.
@_VARIABLE_PATH	Contains the path for variable files, for example, C:\IAUTHOR\VARIABLE.

Database Objects

A database is an organized collection of information that your application can access and use. One application can use several databases and one database can be used by several applications. When do you use a database? When you want your application to efficiently keep track of large amounts of information. Even though the database may be very large, containing information on many customers, students or products for example, only a small piece of it is loaded into memory at one time.

The most powerful way to access a database is through the Database object which is available through the SmartObject Editor. This object allows you to take advantage of the Open Database Connectivity (ODBC) interface. ODBC lets your application use Structured Query Language (SQL) to access a variety of different database formats including:

Access	Oracle
Btrieve	Paradox
dBASE	SQL Server
Foxpro	Sybase

A control bar can provide the user with tools for easy navigation through the database. Records are displayed in Text objects that are bound (associated) with the Database object. Once a record is retrieved the user can perform key operations such as adding, deleting and updating records.

Database Objects use the following properties:

Area
AutoErrorDisplay
AutoNavigate
Bottom
CanAppend
CanUpdate
ClipSiblings
Command
CommandOnCreation
ConnectExclusive
ConnectString
CursorName
DataSources
DeleteProtected
EnableFirst
EnableLast
EnableNext
EnablePrev
EnableUpdate
FamilyName
FieldCound
FieldNames
Height
Layer
Left
Location
NotifyOnRecordFirst
NotifyOnRecordLast
NotifyOnRecordNext
NotifyOnRecordPrev
NotifyOnRecordUpdate
ObjectData
ObjectName
PageName
PositionCurrent
PositionSeek
RecordCount
RecordData
RecordStatus
Rectangle
ResultString
Right
Size

SQLText
Status
TableCount
TableNames
Top
Visible
VisibleFirst
VisibleLast
VisibleNext
VisiblePrev
VisibleUpdate
Width