File Menu

New...

Starts a new SmartObject file. Allows you to start the file in one of three modes: Page Layout, Formatted Text, or Text (ASCII).

Related Topics:

Starting New SmartObject Files
Using Formatted Text Mode
Using ASCII Mode

Open...

Opens an existing SmartObject file.

Related Topic:

Opening Existing SmartObject Files

Save

Saves an existing SmartObject file with the same name.

Related Topic:

Saving Existing SmartObject Files

Save As...

Allows you to save and name a SmartObject file for the first time, or to save and rename an existing SmartObject file.

Related Topic:

Saving New SmartObject Files

Delete

Allows you to permanently delete the current 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.

Print...

Prints one or more pages in the current SmartObject file.

Related Topic:

Printing SmartObject Files

Printer Setup...

Allows you to select a printer driver and a printer connection.

Exit

Exits the SmartObject Editor. Lets you save unsaved changes before quitting.

Related Topic:

Quitting the SmartObject Editor

Edit Menu

Cut

Removes the selected object or text. A copy of the removed item is placed on the Clipboard.

Related Topics:

Cutting Objects
Cutting Text

Copy

Copies the selected object or text onto the Clipboard.

Related Topics:

Duplicating Objects
Copying Text

Paste

Inserts the contents of the Clipboard onto the current page.

Related Topics:

Duplicating Objects
Pasting Text

Clear

Clears the selected text or object. Cleared items are permanently removed and are *not* placed on the Clipboard.

Related Topics:

Clearing Objects
Clearing Text

Copy To...

Copies the selected text in a Text object to a specified ASCII text file. If that file existed previously, its previous contents are overwritten.

Paste From...

Inserts the contents of an ASCII text file into a Text object.

Related Topic:

Creating Text Files

Page Menu

New Page...

Allows you to create a new page in the current SmartObject file. Allows you to create a new page in the current SmartObject file.

Page Maintenance...

Displays a the Page Maintenance dialog box which lets you: create a new page, rename a page, copy or delete a page, or import a page from another SmartObject file.

Related Topics:

Creating New Pages
Copying Pages
Deleting Pages
Importing Pages
Changing Pages
Renaming Pages

Page Properties

Displays a cascading menu of commands that let you set certain characteristics of the current page.

Color...

Allows you to set the color of the current page.

Related Topic:

Working with Page Color

Page Name...

Allows you to name the current page.

Effect...

Allows you to select a special display effect for the page background.

Next

Displays the next page in the SmartObject file.

Previous

Displays the previous page in the SmartObject file.

Go To...

Allows you to jump to a particular page in the file.

Objects Menu

Tools

Displays a cascading menu of tools that let you select or create different kinds of objects.

Select Object

Causes the Select Object tool to be in effect. Depending on whether you select one object or a group of objects, different editing tasks become available. For example, you can cut, copy, paste or resize one selected object. You can move or align multiple objects or set a common property such as FamilyName.

Related Topic:

Selecting and Editing Objects

Audio

Allows you to create objects that play wave files, MIDI files, or CD selections.

Related Topic:

Audio Objects

Button

Allows you to create button objects such as push buttons, check boxes, and radio buttons.

Related Topic:

Button Objects

Combo Box

Allows you to create Combo Box objects such as drop-down, drop-down list, and simple boxes.

Related Topic:

Combo Box Objects

Database

Allows you to create a Database object. The Database object lets your application use SQL commands to access a database via Microsofts ODBC interface.

Related Topic:

Database Objects

Graphic

Allows you to create graphic objects. Once you draw a graphic object you can set its properties, for example, so that a particular graphic is displayed.

Related Topic:

Graphic Objects

IconAnimate

Allows you to play an animation file created with IconAnimate.

Related Topic:

IconAnimate Objects

Keyboard

Allows you to create Keyboard. Define a list of keys that the user can activate to generate an event.

Related Topic:

Keyboard Objects

List Box

Allows you to create List Box objects. Once you draw a List Box object you can set its properties, for example, so that a particular list of items is displayed.

Related Topic:

Listbox Objects

Menu

Allows you to create Menu objects that are top-level or floating pop-up menus.

Related Topic:

Menu Objects

Movie

Allows you to create Movie objects that play digital video files and third-party animation clips.

Related Topic:

Movie Objects

OLE

Allows you to create OLE 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 and create data that eventually becomes an object within your IconAuthor application.

Related Topic:

OLE Objects

Text

Allows you to create text objects. Once you draw a text object you can set its properties, for example, so that it is editable. If a text object is editable, a user can type in it at runtime. If a text object is not editable, the author types text in it within the SmartObject Editor and the information is display-only at runtime.

Related Topic:

Text Objects

Timer

Allows you to create timer objects. Once you draw a timer you can set its properties, for example, so that it is set to count down, count up, go off periodically, or go off at a particular time of day.

Related Topic:

Timer Objects

Transparent

Allows you to create transparent objects. A transparent object acts like a push button that cannot be seen. The transparent object can cover a small portion of a page or an entire page.

Related Topic:

Transparent Objects

Variable

Allows you to create Variable objects. A Variable object lets you load one or more variables into memory.

Related Topic:

Variable Objects

Object Properties

Displays a cascading menu of commands that let you set the properties for the currently selected object. The items in this menu vary depending on the object that you select. (The items in this menu are identical to the items that appear in the pop-up menu when you right mouse click on an object.)

Properties...

Allows you to display the Properties dialog box for the selected object. The list of properties varies depending on the object that you select.

Related Topic:

Setting Object Properties

The following items in the Object Properties menu are organized alphabetically below. This is not how they appear in the SmartObject Editor.

1 1/2

This menu item is only available when you are editing text. Allows you to one and one-half space the text in the current paragraph.

All Hotwords

This menu item is only available when you are editing text. Makes every word in the object a Hotword.

Related Topic:

Using Hotwords

Apply Hotwords...

This menu item is only available when you are editing text. Adds a hotword file to a text object or FTT file.

Related Topic:

Applying a Hotword File

Block Styles...

This menu item is only available when a Text object is selected. Allows you to specify the certain characteristics of the object such as colors, shading, and shadow.

Related Topic:

Changing Block Style

Button Styles...

Allows you to specify the style of the object: Push Button, Icon Button, Check Box, Radio Button, or Group Box. By default, the property list for a Button object is for Push Button style. If you change the style the property list also changes.

Related Topic:

Button Objects

Center

This menu item is only available when you are editing text. Allows you to center the text in the current paragraph.

Combo Box Styles...

Allows you to specify the style of a Combo Box: drop-down, drop-down list, or simple.

Related Topic:

Combo Boxes

Convert Data to Static...

Allows you to convert the *data* in an OLE object to static. When the data is static it means that it cannot be changed because the connection to the server has been permanently severed. Static data is different from a static object (an object that is not live). An OLE object with static data can be live and still have its properties, such as visible and enabled, changed at runtime.

Related Topic:

Permanently Breaking the Server Connection

Database Styles...

Allows you to choose between two styles, Iconic and Control Bar. If you choose Control Bar, the object appears as a row of buttons the user can manipulate to move through the database. If you choose Iconic, the object is not visible at runtime and you must build an icon structure that displays and manipulates an alternative set of custom control buttons.

Related Topic:

Database Styles

Double

This menu item is only available when you are editing text. Allows you to double space the text in the current paragraph.

Fill Color...

This menu item is only available when you are editing text. Allows you to change the color used for the background of text.

Fonts...

This menu item is only available when you are editing text. Allows you to change the font information (such as font type, size and color) for text.

Hotword

This menu item is only available when you are editing text. Allows a string of text to be designated as selectable at run time. Subsequently, the selection of that text triggers an event.

Related Topic:

Using Hotwords

Hotword Data...

This menu item is only available when you are editing text in a Text object. Allows you to assign an index number to a hotword. It also allows you to select a Hotword text style.

Related Topic:

Setting Indexed Hotwords

IconAnimate Styles...

Lets you choose between playing the animation as part of the background or as a window.

Related Topic:

IconAnimate Objects

Import File...

Lets you import the contents of an ASCII text file into a Variable object. This item is only available when you double-click on the object to access text entry mode.

Related Topic:

Variable Objects

Input Styles...

This menu item is only available when a Text object is selected. Allows you to specify the kind of characters that can be entered in an object (by the author and/or the user) and in some circumstances, the positioning and quantity of text allowable.

Related Topic:

Setting Input Style

Insert New Object...

Allows you to insert data (from an OLE server) into the currently selected OLE object.

Related Topic:

OLE Objects

Item List...

Allows you to specify the items that you want to appear in a list.

Related Topic:

Entering Data in List Boxes

Left

This menu item is only available when you are editing text. Allows you to left align the text in the current paragraph.

Links...

This menu item is only available for OLE objects with which you have previously used the Paste Link... command. The Links... command displays the Links dialog box that describes the current status of the OLE link, such as the server name and filename.

Related Topic:

OLE Objects

Menu Design...

Allows you to specify the menu headings and menu items (commands).

Related Topic:

Menu Objects

Menu Styles...

Allows you to specify the style of a menu: top level or floating pop-up.

Related Topic:

Menu Objects

Object

Allows you to exercise the actions associated with the object.

Object Type...

Allows you to indicate whether an object is live or static. When an object is live its properties can change at runtime. Otherwise, it behaves as if it is an unchanging part of the background of the display. (This item only appears for a limited number of objects because all other objects are *always* live and cannot be made static.)

Related Topic:

Making Objects Live or Static

Paste

For OLE, allows you to paste data (such as a picture or selected cells from a spreadsheet) that was previously cut or copied to the Clipboard (from a server application) into your OLE object. For Graphic objects, allows you to paste an image (previously cut or copied to the Clipboard) into a Graphic object.

Paste Link...

Allows you to paste server data from the Clipboard into an OLE object. The data is linked to the file from which it was originally copied. As a result, any time that you open the server application and change and save that file, the changes you make are automatically updated in the OLE object on the SmartObject page.

Related Topic:

OLE Objects

Remove Hotwords

This menu item is only available when you are editing text. Removes a hotword file from a text object or FTT file.

Right

This menu item is only available when you are editing text. Allows you to right align the text in the current paragraph.

Single

This menu item is only available when you are editing text. Allows you to single space the text in the current paragraph.

Sound Styles...

Allows you to specify the style: Wave Button, Wave, CD Button, CD, MIDI Button, MIDI.

Related Topic:

Audio Objects

Text Color...

This menu item is only available when you are editing text. Allows you to change the color for text.

Text Styles...

This menu item is only available when you are editing text. Allows you to work with the named styles used to set the characteristics of text.

Timer Styles...

Allows you to specify the style of an object: Periodic, Count Down, Count Up, or Alarm.

Related Topic:

Timer Objects

Variable Icon

When this option is checked, the object appears as an icon. (The icon is not visible at runtime.)

Related Topic:

Variable Objects

Variable Window

When this option is checked, the object appears as a window. Double-click on the window to access text entry mode. (The window is not visible at runtime.)

Related Topic:

Variable Objects

WordWrap On

When this option is checked, it causes text to automatically wrap to the next line when it reaches the left border of the Variable window.

Related Topic:

Variable Objects

Alignment

This menu item is only available when a group of objects is selected. Displays a cascading menu of commands that let you change the alignment of the objects in the group.

Left

All objects are moved so that their left side is aligned with the left-most selected object.

Right

All objects are moved so that their right side is aligned with the right-most selected object.

Top

All objects are moved so that their top border is aligned with the top-most selected object.

Bottom

All objects are moved so that their bottom border is aligned with the bottom-most selected object.

Center Vertical

All objects are moved so that they are centered on a common vertical axis.

Center Horizontal

All objects are moved so that they are centered on a common horizontal axis.

Order

This menu item is only available when an object is selected that is at least partially obscuring one or more other objects. Displays a cascading menu of commands that let you change the position of the object in the stack.

Layers...

Displays a dialog box that allows you to manipulate the layering order of objects. For example, you can move an object backward or forward.

Tab Stops...

Displays a dialog box that allows you to change the tab order of objects that can be tab stops. Changing

the tab stop order changes the layering of objects.

Options Menu

Path File...

Causes the SmartObject Editor to read the .PTH file of an IconAuthor application. The .PTH file tells the SmartObject Editor where the files for that application are located.

Related Topic:

Path Information

Full Screen

When this option is toggled on, the SmartObject Editor appears full screen. (The menus, ribbon bar and status bar appear in a floating window that can be moved around the screen.)

Hide Tools

This option is available when the SmartObject Editor appears full screen. It allows you to hides the menu bar (and the ribbon and status bars). To make the menu bar visible again press CTRL + T.

Grid Alignment...

Displays the Grid dialog box. The Grid dialog box lets you toggle the grid feature on and off, make the grid visible or not visible, and change the cell dimensions used for the grid.

Related Topic:

The Grid

Confirm Clear

When this option is toggled on, it causes the SmartObject Editor to ask you to confirm whether you want to clear an object when you use the Clear command from the Edit menu.

Overlay Mode

When this option is toggled on the current video
frame is displayed on the screen. (This feature is only available if the appropriate video hardware has been installed and set up.)

Related Topic:

Displaying Video in the SmartObject Editor

Ribbon Bar

When this option is toggled on the ribbon bar is visible.

Status Bar

When this option is toggled on the status bar is
visible.

Save Settings On Exit

When this option is toggled on and you exit the SmartObject Editor, settings that describe the condition of the editor (such as the size of the window or the grid settings) are remembered the next time you restart the editor.

Help Menu

Index

Displays an alphabetical list of all Help topics that are available.

Keyboard

Displays the accelerators for performing some tasks with the keyboard instead of the mouse.

Commands

Displays an explanation of commands.

Procedures

Displays a description of how to use IconAuthor

Using Help

Displays a short tutorial and other information about using Windows online Help.

About SmartObject...

Displays SmartObject Editor copyright and version information.

Objects

Each kind of object is called a class. The following object classes are available in the SmartObject Editor:

Audio
Button
Combo Box
Database
Graphic
IconAnimate
Keyboard
List Box
Menu
Movie
OLE
Text
Timer
Transparent
Variable

You include these objects in your file by choosing the appropriate object tool, drawing the object on the page, and then defining the object.

All objects have certain properties (characteristics) that you define in order to make them appear and perform in a particular manner. For example, a button object has a property called Label that allows you to change the text label that appears on the button. Buttons also have a property called Style that lets you specify what kind of button you want, for example, a push button, a radio button, or a check box.

The most important property of an object is whether it is static or live. Many objects can only be live, while Text, Graphic, and OLE objects can be live or static.

Two additional objects are automatically available in every application:

System
Window

Related Topics:

Making an Object Live or Static
Creating Objects
Setting Object Properties

Live Object Maintenance

If you plan to create live objects in your SmartObject files it is important that you understand how to take care of those objects at runtime. For example, once a live object exists, you can use an ObjSet icon in IconAuthor to reset its properties.

Another important consequence of using live objects is that once you create them on the SmartObject page, and display the page in IconAuthor, the objects continue to exist in memory until you deliberately remove them. Once there is no longer a need for an object, you must use an ObjDelete icon to delete it permanently. When you use an ObjDelete icon you remove the object from memory but you don't effect the original object in the SmartObject file. In fact, once an object is deleted at runtime, the only way to recreate it is to display the original SmartObject page again.

Note: You cannot delete the System object or the Window object.

Creating Objects

Use the object tools to draw objects. By default, several of the objects that you draw are initially white-filled with a thin black border. The exceptions are objects such as the timer object that appears as a clock, the button object that appears as a push button, and the transparent object which has horizontal lines across it. By default, when you draw objects they are conforming to an invisible grid in the work area.

When you begin drawing keep in mind that you don't have to draw the object in the precise dimensions or position because you can always select an object and then resize, move, or even delete it. The maximum size of objects is defined by the dimensions of your work area. You should also be aware that drawing objects is typically just the first step in making the objects appear and perform as you want. For example, once you draw a text object you still need to enter text in it and once you draw a button you still need to change its text label so that it says OK, Cancel, etc. to meet the needs of your application.

You can create objects that are independent of the other objects on the page or you can create objects that overlap or completely hide other objects.

To draw an object:

1. Click on the Object Tools button in the function ribbon.

   A menu of the available object classes appears.

2. Click on the tool that represents the object you want to create.

   Note that the first menu option, Select Object, is not a tool for creating objects, but rather for selecting objects. When you select an object tool the cursor changes to a cross-hair.

3. Position the cursor in the work area at the point where you want the upper left corner of the object to appear.

4. Press and hold the left mouse button and drag diagonally down to the right until the cursor reaches the location for the lower right corner of the object.

   Note: If you begin drawing an object and decide not to continue there is an easy way to stop the process. While continuing to press the left mouse button click the right mouse button. The process is canceled.

5. Release the mouse button.

   The object is drawn.

You can continue to create new objects of the chosen class. If you want to draw another class of object, choose a different tool from the menu.

Object Alignment

In many situations, it is important that objects are in precise alignment with one another. For example, you may want two graphics centered on the page or four buttons aligned along the right side of the page. The SmartObject Editor comes with two features to aid in aligning objects: the grid and the Align Objects button.

The Grid

By default, when you draw objects they are conforming to an invisible grid in the work area. The grid makes it easy for you to align objects such as a vertical column of on-screen buttons. You can make the grid visible or invisible. You can also change the dimensions of the grid cells or simply disable the grid altogether. Use the Grid dialog box to change the grid options.

To change grid options:

1. Click on the Set Grid button.

   The Grid dialog box appears.

2. As necessary, make changes to the grid options.

   Grid Size

   By default, the grid lines define cells that are 10 x 10 pixels. Type new numbers in these text boxes to redefine the size of the cells.

   Snap To Grid

   By default, the grid is active. To disable the grid click on the Snap To Grid option to toggle it off.

   Show Grid

   By default, the grid is active but invisible. To make the grid visible click on the Show Grid option to toggle it on. When this option is on, points appear on the screen to identify the location of the lines that define the grid.

3. Choose OK to close the Grid dialog box.

The Align Objects Button

Regardless of whether the grid is in effect, you have the option of using the Align Objects button to align objects. In effect, you draw a box around a group of objects to select them and then click on the Align Objects button to show a menu of alignment options. You choose an option, such as left (to align all selected objects with the left-most selected object) or right (to align all selected objects with the right-most selected object).

To align objects with the Align Objects button:

1. Make sure that the Select Object tool (the arrow cursor) is active by clicking on the background.
2. Press and hold the left mouse button above and to the left of the objects you want to align.
3. Drag down and to the right to draw a box that encloses the objects.
4. Release the left mouse button.

   When you release the left mouse button the box disappears and the objects within the box are selected. (Each object is surrounded by eight small sizer blocks.)

   And, the Align Objects button becomes active in the function ribbon.
5. Click on the Align Objects button.

   A menu of alignment options appears.
6. Choose the desired alignment option and the selected objects are realigned.

The following paragraphs describe the effect created by the different alignment options.

This option:	**Moves all selected objects so that:**
Left	their left side is aligned with the left-most selected object
Right	their right side is aligned with the right-most selected object
Top	their top border is aligned with the top-most selected object
Bottom	their bottom border is aligned with the bottom-most selected object.
Center Vertical	they are centered on a common vertical axis
Center Horizontal	they are centered on a common horizontal axis

Selecting and Editing Objects

You can select one object or several objects. Once you select an object you can edit it. However, different editing tasks are available for single and multiple objects.

To select a single object:

Click on the object.

A selected object is immediately surrounded by eight sizer blocks; one at each corner and one in the center of each side. Event if the object you select is partially obscured by another object the eight sizer blocks appear, defining the selected object even though it is still obscured.

There is only one situation where you cannot just click on an object to select it. If you are editing text in a text object and decide to select that text object you can't just click on it. You have to click on some other object (or the page background) and then click on the text object you were editing.)

To select multiple objects:

1. Make sure that the Select Object tool (the arrow cursor) is active by clicking on the background.
2. Press and hold the left mouse button above and to the left of the objects you want to select.

3. Drag down and to the right to draw a box that encloses the objects.
4. Release the left mouse button.

   When you release the left mouse button the box disappears and the objects within the box are selected. (Each object is surrounded by eight small sizer blocks.)

Related Topics:

Resizing Objects
Moving Objects
Duplicating Objects
Removing Objects
Aligning Objects

Resizing Objects

After you draw an object you usually have the ability to resize it. (Graphic objects are the exception.) Frequently, it's practical to create an object, set some of its properties, and then resize it as necessary.

To resize an object:

1. Select the object.
2. Position the cursor over one of the eight sizer blocks.
3. Press and hold the left mouse button and drag the sizer block to a new position.

   As you drag the sizer block the object is extended (or compressed) in the direction in which you drag.
4. When the object reaches the desired size release the mouse button.

Note: By default, when you resize an object it conforms to an invisible ~~grid~~ in the work area. This feature lets you align objects vertically and horizontally.

Related Topics:

Hints for Resizing Text Objects
Hints for Resizing Graphic and OLE Objects

Resizing Text Objects

Often, you'll find it useful to resize a text object. For example, you may create a text object, begin entering text in it, and find that the object isn't large enough to hold all the characters you want. When you make a text object larger you are creating more space for entering additional text.

When you need to make a text object smaller you have to be careful not to make it too small. For example, if you draw a text object, enter text in it, and then make the text object smaller so that it closely frames the text, you may accidentally *scroll* some characters out of view. You can bring characters back into view by making the object slightly larger and using the arrow keys.

Resizing Graphic Objects

By default, graphic objects are not resizable. You'll find that initially, when you draw one of these objects and select it, you can only move the object, not resize it. This is because of the default settings for the DrawStyle property. By default, the graphic object's DrawStyle is set to Size Graphic. This means that the object is automatically resized to fit the graphic.

You can make graphic objects resizable by changing the DrawStyle property to another value such as Scale Graphic. As an example, when you use Scale Graphic, the graphic is scaled to fit within the object as you drew it. Keep in mind that if you use a property setting like Scale Graphic, resizing a particular dimension of the object may result in distorting the image by stretching or compressing it.

Moving Objects

There are two basic ways to move objects. You can drag one or more objects to another position on the page. Or, if an object is part of a stack of layered objects, where one object appears to be in front or in back of other objects, you can move an object so that it is positioned differently in the layering.

Related Topics:

Dragging Objects
Object Layering

Dragging Objects to a New Position

Once you have selected one object or a group of objects, you can drag to a new position on the page. This is useful because, frequently, you will find that you did not draw an object in precisely the position you want. When you drag an object the information within it, for example, text or a graphic, moves with it.

To drag objects:

1. Select the object or the group of objects.

   The borders of any selected objects are surrounded by eight sizer blocks, one at each corner and one in the center of each side.
2. Position the cursor over the selected object or anywhere over the selected objects.
3. Press and hold the left mouse button and drag the object(s) to the new location.
4. Release the left mouse button to re-draw the object(s) at the new location.

Note: By default, when you move text objects they conform to an invisible grid in the work area. This lets you align objects vertically and horizontally.

Object Layering

Each object that you draw exists in a different layer on the page. The first object you draw on a page is in layer 1. The second object you draw is in layer 2, and so on. The lower the layer number, the "closer" an object appears to the background of the page.

Once you've drawn an object, you can still change its place in the layering order. Within the SmartObject Editor you can use commands to bring it forward (so that it is in front of another object) or you can send it backward (so that it is in back of another object). You can also change the objects layering at runtime via the Layer property.

Important: Changing the layering order of objects can change the Tab Stop order of objects.

To change the layering position of an object:

1. Click on the Object Order button.
2. Choose Layers... from the pop-up menu.

   The Layers dialog box appears.

   The objects on the page are listed from lowest to highest layer. Each object is identified by its ObjectName. The object that was selected when you clicked on the Object Order button is selected. If a page contains unnamed objects, the objects are represented in the Layers dialog box as "[Empty]."
3. Select the object you want to move in the layering order.
4. Position the cursor at the point in the list to which you want to move the object.

   The cursor appears as a special insertion marker.
5. Click on the new position to change the order of the selected object.
6. Choose Close to close the Layers dialog box.

You can also use the command buttons in the Layers dialog box to change the layer order of an object.

Bring to Front **-** Brings the currently selected object to the front of a stack.

Send to Back - Sends the currently selected object to the back of a stack.

Bring Forward - Brings the currently selected object one position forward in a stack.

Send Backward - Moves the currently selected object one position backward in a stack.

Setting Object Tab Stops

Most objects can be set up as tab stops. When an object is a tab stop, the user can select the object (such as a button) via the TAB key on the keyboard. Once the object is selected, the user can activate the object, for example by pressing the RETURN key.

You need to make two basic decisions when you set Tab Stops. First, you must decide which objects will be Tab Stops. Second, you must decide the sequence in which Tab Stops are selected.

Setting the KeyboardTabStop Property

If you want an object to be a Tab Stop you must ensure that the object's KeyboardTabStop property is set to True. If an object does not have this property, it cannot be a Tab Stop.

Setting the Tab Stop Order

The Tab Stop order is the same as the layering order of objects. The object closest to the background is the first Tab Stop, the second object is the second Tab Stop, and so on. Although the SmartObject Editor lets you re-set the Tab Stop order as desired, note doing so also changes the layering order.

To change the Tab Stop order of an object:

1. Click on the Object Order button.
2. Choose Tab Stops... from the pop-up menu.

   The Tab Stops dialog box appears. The objects on the page that have a KeyboardTabStop property are listed from first to last Tab Stop. Each object is identified by its ObjectName. The object that was selected when you clicked on the Object Order button is selected. If a page contains unnamed objects, the objects are represented in the dialog box as "[Empty]."
3. Select the object you want to move in the Tab Stop order.
4. Position the cursor at the point in the list to which you want to move the object.

   The cursor appears as a special insertion marker.
5. Click on the new position to change the order of the selected object.
6. Choose Close to close the Tab Stops dialog box.

Remember, the changes that you make in the Tab Stop dialog box also affect the layering order of the objects.

Duplicating Objects

Duplicating objects is particularly helpful when you want to create a group of objects that share one or more common characteristics. For example, perhaps they all need to be exactly the same size and belong to the same family. (All objects have a Family property. When objects belong to the same family, at runtime an object icon can be defined to act on an entire family.)

To copy an object:

1. Select the object you want to copy.
2. Click on the Copy button or press CTRL + INSERT.

The selected object is copied and stored in the Clipboard. The second step in duplicating an object is to paste it from the Clipboard back onto a page.

To paste an object:

1. Make sure the correct page is showing.

   If you want to paste the object back onto the same page, you don't have to do anything. If you want to paste it onto another page, go to another page.
2. Click on the Paste button or press SHIFT + INSERT.

   The Paste button appears as follows:

   A copy of the object that is stored on the Clipboard is pasted into the upper left corner of the page. The new object is also selected. As necessary, drag the object to a new position.

Removing Objects

You can cut or clear objects to remove them. When you cut an object it is removed to the Clipboard. Once it is on the Clipboard you can paste it back onto a page to recover it. In fact, you can paste it as many times as you want. When you clear an object it removes the object permanently.

Cutting Objects

When you cut a selected object from the current page a copy of the object is stored in the Clipboard. Optionally, you can paste the object back onto the page, or another page, as many times as you want. Remember that an object remains on the Clipboard until you copy or cut another object, or until you quit the SmartObject Editor.

To cut an object:

1. Select the object to be cut.
2. Click on the Cut button or press SHIFT + DEL.

   The Cut button appears as follows:

   The selected object is deleted and stored on the Clipboard. If necessary, click on the Paste button to paste the object back into the upper left corner of the page.

Clearing Objects

When you clear a selected object from the current page the object is removed permanently. Use care when you clear objects because they are not recoverable. When you clear an object, it is not stored in the paste buffer and no longer exists on the page.

To clear an object:

1. Select the object(s) to be cleared.
2. Choose the Clear command from the Edit menu.

   A default setting causes a message to appear asking you to confirm that you want to clear the selected object.
3. Choose Yes to clear the object or No to Cancel.

   When you choose Yes, the selected object(s) is deleted.

Optionally you can turn off the message that confirms the Clear action by choosing Confirm Clear from the Options menu

Setting Object Properties

All objects have certain properties that you define in order to make the objects appear and perform in a particular manner. Each object has several properties some of which are common to other objects. Most properties are set to a default. There are two ways to set properties. You can set properties in the SmartObject Editor. You can re-set properties (for live objects only) in IconAuthor (at runtime) via an ObjSet icon.

To set a property in the SmartObject Editor:

1. Click on the object with the right mouse button to open the pop-menu..
2. Choose Properties...

   A Properties dialog box appears with a list box that identifies all of the properties available for the currently selected object. The first property is highlighted by default. The text box at the top of the dialog box contains the value(s) currently set for the highlighted property.
3. Click on the property you want to set (or use the arrow keys).

   The current value (if any) appears in the text box.
4. Change the current setting by typing a new value in the text box or by choosing an option from the drop-down list.

   To type in the text box, either click in it and type (or press the Tab key to change the highlight to the text box and then type).

   To select an option from the drop-down list, click on the down-arrow and click on the desired option (or press the down arrow key to open the list, use the up and down arrow keys to change the selection, and press Return to choose an item).

   The drop-down list box options vary from one object to another and from one property to another. The list options are either explicit values (such as True or False) or they display a dialog box (such as the Color dialog box) to allow you to select a value.

   As soon as you activate the text box by clicking in it (or by tabbing to it) or by opening the drop-down list box, the X and √ buttons are activated. If you type or select a value and then change your mind before you move to another property, you can click on the X to re-display the original value.
5. As necessary, when you finish working with one property you can move to another and change its setting as well.
6. When you are satisfied with the property settings double-click on Control menu box in the top left corner of the Properties dialog box.

Keeping the Properties Box Open for Multiple Objects

You can open the Properties dialog box through one object and then keep it open while you move on to change the properties of other objects. To keep the dialog box open, use the push pin button near the top left corner of the dialog box.

Click on the push pin button to "pin" the dialog box open. The button changes, appearing to be pushed in.

You can now click on another object to set its properties. As soon as you click on another object, the Properties dialog box (its title bar and contents) change to reflect the new object. The Properties dialog box remains open until you click on the push pin to "un-pin" the dialog box and double-click on the Control menu box in the top left corner.

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.

Related Topics:

Audio Object Appearance
Audio Object Interactivity
Playing Wave and MIDI Audio Files
Playing CD Audio
Audio Object Properties

When you right mouse click on an Audio object you can choose the Audio Styles... option to show the following list of available styles:

Wave
CD
MIDI

After you select the desired style, choose OK to close the dialog box. When you subsequently open the Properties dialog box, you will find that the selection of properties varies depending on the style of button you have chosen.

Audio Object Properties

The following list shows all of the Audio object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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

Audio Object Appearance

At runtime, Audio objects are not visible. You can create Button objects that are Push Button or Picture Push Button style and set them up to play the specified audio when the user clicks on them. Or you can have the Audio object play via ObjSet icons without any interaction from the user.

Audio Object Interactivity

There are two basic ways a user can interact with an Audio object. You can use the objects CommandOnCreation and Command properties to play the object. Or, you can use a Button object to let the user play the audio.

The User Clicks on a Button object to Play Audio

You can set up a Push Button or Picture Push Button to play the specified audio (a file or CD track) when the user clicks on it. Your application does not require icons to support this functionality. As soon as the SmartObject page displays the button, the user can click on it. The audio plays until it reaches the end of the media, until the object is deleted, or until the user clicks on the button again.

There are two properties that need to be set to connect a Push or Picture Push Button to an Audio object. The two properties are ControlObjectName and ControlCommand. Set the ControlObjectName property to the ObjectName of the Audio object and set the ControlCommand to the appropriate audio command.

For example, you set the ObjectName property of an Audio object to Audio1. You then want a Push Button object to control the playing of the Audio object. Set the ControlObjectName property of the Push Button to Audio1. Set the ControlCommand property of the Push Button to Open. The Open command pre-loads the audio file. When the user clicks on the Push Button, the audio file will play.

Playing Wave and MIDI Audio Files

Use the FileName property to specify the file you want the object to play. By the default, the PlayCount property, which specifies how many times to play the file, is set to 1. The PositionStart property sets the start point (in milliseconds) from which audio will play. The PositionEnd property sets the end point.

The CommandOnCreation property controls how the object behaves when it first displays. For example, the default is None which means the object does nothing. The Play command causes the file to play immediately. At runtime, set the Command property to further manipulate the object, for example by playing, pausing, or stopping the audio.

At runtime an ObjSet icon can also use the PositionSeek property to specify a desired position in the file. Once you set the PositionSeek property, you can set the Command property to carry out the seek operation. Your application can also use another runtime property, called PositionCurrent, to get the current position in the audio file.

All Wave and MIDI objects have a NotifyonComplete property which generates an event when the audio completes and the NotifyOnError property which generates an event when an MCI error occurs. The resulting error is the current setting of the ResultString property.

Playing CD Audio

The CD objects also use the PlayCount property to specify how many times to play the specified audio. The PositionStart property sets the start point (in Track:Minutes:Seconds:Frames format) on the CD from which audio will play. The PositionEnd property sets the end point. By default, the CD plays to the end of the media.

At runtime, your application can get the current settings for the TrackCount, TrackLength, and TrackNumber properties. As an example, once your application retrieves these values, it can redisplay them on the screen to emulate a CD player.

Like the Wave and MIDI style objects, the CD objects also use the CommandOnCreation property to control how the object behaves when the SmartObject page is displayed. For example, the default is None which means the object does nothing. The Play command causes audio to play immediately. At runtime, set the Command property to further manipulate the object, for example by seeking a position on the disk, playing, or stopping the audio.

All CD objects have a NotifyonComplete property which generates an event when the audio completes and the NotifyOnError property which generates an event when an MCI error occurs. The resulting error is the current setting of the ResultString property.

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

After you select the desired style, choose OK to close the dialog box. When you subsequently open the Properties dialog box, you will find that 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.

Related Topics:

Push Button Appearance
Push Button Interactivity
Push Button Properties

Push Button Appearance

Use the LabelType property to control whether text or a graphic appears in a Push Button. This property can be changed at runtime via the ObjSet icon. If you choose text as the LabelType, use the Label property to set the text that appears in the Push Button. The Font property controls the character font and size. You cannot change the background color or text color of a Push Button.

If you choose graphic as the LabelType, use the FileName property to enter the name of the graphic file that appears. You can also choose to enter a disabled state graphic filename in the FileNameDisabled property.

Push Button Interactivity

There are several basic ways in which a user can interact with a Push Button. Typically, a user generates an event by clicking on a Push Button. You can set up the object so that the structure evaluates the event and branches accordingly. Or you can set up the object so that is associated with another object that executes a command as soon as the user clicks. For example, you can associate a Push Button with an Audio object that contains a Play command. As soon as the user clicks on the Push Button, the Audio object executes the command.

You can also set up Push Buttons so that it is cursor-sensitive. When an object is cursor-sensitive, a user generates an event by moving the mouse cursor onto and off of the object. This is called entering and leaving. For example, you can set up your application so that as the user moves the cursor over (enters) a button, a status bar at the bottom of the screen displays the purpose of the button. When the user moves the cursor off (leaves) the button, the status bar message disappears.

The third way a user can interact with a Push Button is by setting focus to it. The user can do this by either clicking on the object, accessing it via the keyboard or tabbing to it. This works similarly to moving the mouse cursor onto and off of the object.

Note: The following objects can also behave like Push Buttons: Picture Push Buttons, Check Boxes, Radio Buttons, live Text objects, live Graphic objects, and Transparent objects.

Related Topics:

Letting the User Click on a Push Button
Using a Push Button to Control another Object
Making the Object Cursor-Sensitive

Letting the User Click on an Object

You can use the NotifyOnClickLeft, -Middle, and -Right properties to allow a user to click the left, middle, or right mouse button on an object, and record and evaluate the user's action.

Related Topics:

Setting up Active Mouse Buttons
Detecting which Mouse Button was Used
Detecting which Object was Chosen

Setting up Active Mouse Buttons

Each of the NotifyOnClickLeft, -Middle, and -Right properties corresponds to a mouse button. For example, to make the left mouse button allowable, set NotifyOnClickLeft to True. Conversely, if you do not want anything to happen when the user clicks a particular mouse button, set the corresponding property to False. When a user clicks on an object using an allowable mouse button, an event occurs.

Event

An event is an action that is recognized by an application. Every event has a name. For example, the name of the event generated when the user clicks the left mouse button is ClickLeft. (The text that follows "NotifyOn" is always the name of the corresponding event.) After an application displays a SmartObject page, the structure typically contains an ObjEvent icon that waits for an event to occur. Whenever an event occurs, the name of the event is stored in @_Object_Event and the ObjectName of the object that was acted upon is stored in @_Object_Name.

Detecting which Mouse Button was Used

If you want your application to take a different path depending on the mouse button employed, you set more than one Notify- property to True. As mentioned previously, whenever an event occurs, the name of the event is stored in @_Object_Event. For example, if the user can click either left or right on an object and opts to click right, the string "ClickRight" is stored in @_Object_Event. Once the event is stored, a Branches composite of If icons can evaluate the contents of @_Object_Event and branch accordingly.

Detecting which Object was Chosen

You may want your application to take a different path depending on the object that was clicked. Whenever an event occurs, an ObjEvent icon can store the name of the object in @_Object_Name. For example, a user can click on one of two objects that have the ObjectNames Obj1 and Obj2. If the user opts to click on Obj1, the string "Obj1" is stored in @_Object_Name. Once the object name is stored, a Branches composite of If icons can evaluate the contents of @_Object_Name and branch accordingly.

Using a Push Button to Control another Object

You can set the ControlObjectName and ControlCommand properties to control audio, database, movie or variable objects. Set the ControlObjectName property to the ObjectName of the object. Set the ControlCommand property to the appropriate command of the object.

For example, to set up a Push Button so the user can click on it to play the file of an Audio object named Audio1, set the ControlObjectName property to Audio1. Set the ControlCommand property to Open. The audio file will be open and ready to play when the user clicks on the Push Button.

Making the Object Cursor-Sensitive

You can use the NotifyOnEnter and NotifyOnLeave properties to cause an action to occur when the user moves the cursor over an object, and when the user moves the cursor off of an object. If you set NotifyOnEnter to True, an event called "Enter" occurs when the cursor is over the object. If you set NotifyOnLeave to True, an event called "Leave" occurs when the cursor moves off of the object.

Push Button Properties

The following list shows all of the Push Button object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Related Topics:

Picture Push Button Appearance
Picture Push Button Interactivity
Picture Push Button Properties

Picture Push Button Appearance

Use the FileName property to specify the graphic that appears in a Picture Push Button. You can then control how your image appears by using the ButtonStates property. The ButtonStates property lets you choose from three states for the buttons appearance. The three states are: Up, UpDown, and UpDownDisabled. You must have an image for each state if you choose UpDown or UpDownDisabled.

All images must be of equal size and must be contained, side-by-side, in one file. If you set the ButtonStates property to UpDownDisabled, when the button first displays it will appear in the Up state. When the user clicks on the button, it will appear in the Down state. If you choose to set the Enabled property to false, the button will appear in the Disabled state.

Picture Push Button Interactivity

Users interact with Picture Push Buttons in the same way that they interact with Push Buttons.

Related Topics:

Letting the User Click on a Push Button
Using a Push Button to Control another Object
Making the Object Cursor-Sensitive

Picture Push Button Properties

The following list shows all of the Picture Push Button object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Related Topics:

Check Box Appearance
Check Box Interactivity
Check Box Properties

Check Box Appearance

If you want a check box to be on by default, set its Checked property to True. Use the Label property to control the text that appears in the object and use the Font property to control the character font and size. The ColorText property sets the text color and the ColorBackground property sets the color for the face of the button.

Check Box Interactivity

There are several basic ways in which a user can interact with a Check Box. The first two ways are identical to a Push Button.

How Check Boxes Peform the Same as Push Buttons:

Letting the User Click on a Push Button (or Check Box)
Making an Object Cursor-Sensitive

Delayed Evaluation of Check Boxes

You can set up a Check Box to let the user interact but not generate an immediate result. Often an application presents a number of Check Boxes (and Radio Buttons, List Boxes, etc.) at once, as in a dialog box. The user sets set several options without causing an immediate action. The user then indicates that he or she is finished setting options, for example, by clicking a Push button such as OK or Cancel. The application evaluates the state of the Check Boxes and other options as the user set them and branches accordingly.

Related Topic:

Delayed Evaluation of Check Boxes

Check Boxes that are Bound to Database Objects

Check Boxes (along with Radio Buttons, Combo Boxes, List Boxes and Text objects) can be bound to a Database objects recordset. This means you can use a Check Box to display specific information about a record in the recordset. For example, you have a recordset of all the students that have completed your course. Each record has a field that tells you if the student passed or failed the course. To display this information, you could bind two Check Box objects to your Database object. One Check Box object is labeled Passed, the other object is labeled Failed.

This information is updated as the user looks at each students record. As soon as the user displays a record for a particular student, either the Passed or Failed check box is automatically checked. You can also set up Check Boxes so that the user can check or uncheck them and then update the record to the recordset.

Related Topic:

User Interaction with Bound Check Boxes

Delayed Evaluation of Check Boxes

To let the user set a Check Box and not cause an immediate result (an event), set the object's NotifyonClickLeft, -Middle, and -Right properties to False. The user can still turn the object on and off by clicking the left mouse button, because the switching feature of a Check Box is always available (unless the object is disabled). Note that clicking the right or middle mouse button never affects the on/off state of a radio button.

At runtime, when your application needs to know the state of the Check Box, use an ObjGet icon to learn whether the object is on or off. The ObjGet icon can retrieve the current setting of the object's Checked property. If the Checked property is set to True, the object is checked. False means that it is unchecked. Your application can evaluate the setting and branch accordingly.

User Interaction with Bound Check Boxes

To bind a Check Box to a database recordset, you need to set the DataFieldName property to the name of the database field you want it to display. You also need to set the DataObjectName to the name you gave the Database object that is manipulating the database. Use the DataValueChecked property to specify the value a field must match in order to check the button. You also need to set the DataValueUnChecked property to specify the value a field must match in order to uncheck the button. For more information, see the Database Object

Check Box Properties

The following list shows all of the Check Box object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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, the three Radio Buttons to the right 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.

Related Topics:

Radio Button Appearance
Radio Button Interactivity
Radio Button Properties

Radio Button Appearance

If you want a Radio Button to be on by default, set its Checked property to True. Use the Label property to control the text that appears in the object and use the Font property to control the character font and size. The ColorText property sets the text color and the ColorBackground property sets the color for the face of the button.

Radio Button Interactivity

There are several basic ways in which a user can interact with a Radio Button. The first two ways are identical to a Push Button.

How Radio Buttons Peform the Same as Push Buttons:

Letting the User Click on a Push Button (or Radio Button)
Making an Object Cursor-Sensitive

Delayed Evaluation of Radio Buttons

You can set up a Radio Button to let the user interact but not generate an immediate result. Often an application presents a number of Radio Buttons (and Check Boxes, List Boxes, etc.) at once, as in a dialog box. The user sets set several options without causing an immediate action. The user then indicates that he or she is finished setting options, for example, by clicking a Push Button such as OK or Cancel. The application evaluates the state of the Radio Buttons and other options as the user set them and branches accordingly.

Related Topic:

Delayed Evaluation of Radio Buttons

Radio Buttons that are Bound to Database Objects

Radio Buttons (along with Check Boxes, Combo Boxes, List Boxes and Text objects) can be bound to a Database objects database file. This means you can use a Radio Button to display specific information about a record in the database file. This information is updated as the user looks at different records. The user can also use the Radio Buttons to change information in a record and then update the record to the recordset.

Related Topic:

User Interaction with Bound Radio Buttons

User Interaction with Bound Radio Buttons

To bind a Radio Button to a database recordset, you need to set the DataFieldName property to the name of the database field you want it to display. You also need to set the DataObjectName to the name you gave the Database object. Use the DataValueChecked property to specify the value a field must match in order to turn the button on. For more detailed information, see the section on the Database Object. For more information, see the Database Object.

Delayed Evaluation of Radio Buttons

To let the user set a Radio Button and not cause an immediate result (an event), set the object's NotifyOnClickLeft, -Middle, and -Right properties to False. The user can still turn the object on and off by clicking the left mouse button, because the switching feature of a Radio Button is always available (unless the object is disabled). Note that clicking the middle or right mouse button never affects the on/off state of the Radio Button.

At runtime, your application uses an ObjGet icon to determine which Radio Button in a group is on. The ObjGet icon retrieves the CheckedRadioButton property for the family of Radio Buttons and stores the name of the button in a variable. A Branches composite can evaluate the name of the chosen object and branch accordingly.

Radio Button Properties

The following list shows all of the Radio Button object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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 buttons. The figure on the right shows how a Group Box labeled Colors is used to group together three Radio Buttons and a Check Box that all allow the user to make color changes.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 buttons in a visually helpful way.

Group Box Properties

The following list shows all of the Group Box object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Related Topics:

Combo Box Styles
Combo Box Appearance
Entering Items in List Boxes
Combo Box Interactivity
How Combo Boxes Work with Database Objects

Combo Box Styles

Click the right mouse button on a Combo box object and choose Combo Box Styles... from the pop-up menu to display the choices for the different kinds of buttons.

Drop-down style
Drop-down list style
Simple style

Select the desired style and choose OK to close the dialog box. When you subsequently open the Properties dialog box, you will find that the selection of properties varies depending on the style you have chosen.

Combo Box Appearance

When you first draw a Combo Box, you draw an object that includes the area that the text box and the list box will occupy. The horizontally-lined area below the text box indicates the area of the list box.

You determine the size of the text box and list box (open or closed) when you draw the object. If there are too many items to all be visible at once, the list box will have a scroll bar. The items in a list box will be sorted alphabetically if you set the object's Sort property to True.

Use the Font and ColorText properties to control the font and color of characters in the Combo Box. The ColorBackground sets the color of the area behind the text. By default, the text box is empty. You can set it to a specific item via the SelectedItemNumber property.

Drop-Down Style Combo Boxes

The drop-down style Combo Box has a text box in which the user can type a value and a drop-down list from which the user can select an item.

A user can edit the value in the text box or open the list box and select an item. When the user clicks on the down arrow that is detached and to the right of the text box (or presses the F4 key), a drop-down list of items appears. The user can click on an item to select it, or use the up and down arrow keys to change the highlighted selection.

If the user decides to type a value into the text box, the list box attempts to find and highlight a match (in the list) to the characters the user types.

Note: The user can use the keyboard to make a selection from the list box without actually opening it. When the user presses the up and down arrow keys, the value in the text box changes as it cycles through the hidden list.

Drop-Down List Style Combo Boxes

The drop-down list style Combo Box has a display-only text box and a drop-down list from which the user can select an item.

When a user clicks on the down arrow attached to the right side of the text box (or presses the F4 key), a drop-down list of items appears. The user can click on an item to select it, or use the up and down arrow keys to change the highlighted selection.

Remember, the user *cannot* type in the text box in a Combo Box that is drop-down list style.

Note: The user can use the keyboard to make a selection from the list box without actually opening it. When the user presses the up and down arrow keys, the value in the text box changes as it cycles through the hidden list.

Simple Style Combo Boxes

The simple style Combo Box has a text box in which the user can type a value and an open list box from which the user can select an item. The following figure shows an example of how a drop-down Combo Box might appear at runtime.

If the user decides to type a value into the text box, the list box attempts to find and highlight a match to the characters the user types.

The user can type in the text box or use the list box to select a value. The user can click on an item to select it, or use the up and down arrow keys to change the highlighted selection.

Combo Box Interactivity

There are two basic ways in which a user can interact with a Combo Box. Normally, a user generates an event by making a final selection from the Combo Box, by changing the currently highlighted item, or by typing a new value in the text box (if its editable) and pressing Return. These actions can generate events that can be evaluated by your application. You can also set up a Combo Box so that a user generates an event simply by moving the mouse cursor on to and off of the object. For example, you can set up your application so that as the user moves the cursor over a Combo Box, a status bar at the bottom of the screen displays the purpose of the object. When the user moves the cursor off the object, the status bar message disappears.

Related Topics:

Detecting when the User Makes a Final Selection
Detecting When the User Changes the Highlighted Selection
Learning What the User Selected or Highlighted
Making the Object Cursor-Sensitive
How Combo Boxes Work with Database Objects

Detecting when the User Makes a Final Selection

Depending on the style of the Combo Box, a user can make a final selection in the following ways:

+ Type in the text box and press Return.
+ Click on a list item.
+ Use the arrow keys to change the highlighted list item and press Return. (If the list box is not open, this operation still cycles through the hidden list box items.)

If you set the NotifyOnSelect property to True, when a user makes a final selection from a Combo Box a "Select" event is generated. Your application can include an ObjEvent icon to await the user's interaction. When the user makes a final selection, the event "Select" is stored in @_Object_Event and name of the affected object is stored in @_Object_Name. A Branches composite can test these values to detect when a user makes a selection and/or from which Combo Box the selection was made.

Detecting When the User Changes the Highlighted Selection

The user can change the highlighted selection in a Combo Box's list box by pressing arrow keys. (If the list box is not open, this operation still cycles through the hidden list box items.)

If you set the NotifyOnSelectChange property to True, when a user changes the highlighted selection in a Combo Box a "SelectChange" event is generated. Your application can include an ObjEvent icon to await the user's interaction. When the user makes a final selection, the event "SelectChange" is stored in @_Object_Event and name of the affected object is stored in @_Object_Name. A Branches composite can test these values to detect when a user changes the highlight and/or in which Combo Box the action occurred.

Learning What the User Selected, Highlighted, or Typed

When a user makes a final selection or changes the highlighted selection in a Combo Box, the object's SelectedItemData property is reset to the user's selection. Once your application detects that a "Select" or "SelectChange" has occurred, an ObjGet icon can retrieve the current setting of the SelectedItemData property and store it in a variable. The application can then take action based on the user's selection.

Similarly, if the user types in a Combo Box and presses Return, the objects TextBoxData property is set to the value the user typed. The pressing of the Return key generates a Select event. An ObjGet icon can retrieve the current setting of the TextBoxData property and store it in a variable.

How Combo Boxes Work with Database Objects

Combo Boxes (along with Check Boxes, Radio Buttons, List Boxes and Text objects) can be bound to a Database object. When the Database object locates a record, the Combo Box is automatically set to the appropriate value in the list.

For example, consider a database that contains sales information. The database contains fields such as Salesperson ID and Region. The application uses a Database object to open the database. It also includes a Combo Box whose list has four options: Region 1, Region 2, Region 3 and Region 4. The Combo Box is bound to the Database object and is set up to display the Region value for the current database record.

As soon as the Database object finds a record, the Combo Box is automatically set to Region 1, Region 2, Region 3, or Region 4, whichever is appropriate for the current record. The user can leave the setting as is, make a different selection from the list, or type a new value (such as Region 5). The new selection is updated in the database as soon as another record becomes current.

To bind a Combo Box to a Database object, you need to set the DataFieldName property to the name of the database field you want it to display. You also need to set the DataObjectName to the name you gave the Database object.

Combo Box Properties

The following list shows all of the Combo Box properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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

Database Objects

The Database object lets you work with database files via Microsofts ODBC (Open Database Connectivity) interface. ODBC is an industry standard that provides common Structured Query Language (SQL) access to a variety of Database file formats. SQL is a standard database language that lets you define, manipulate and manage data.

Use a third party Database software package to create a database file. You must install the appropriate ODBC driver for the database software package and add the database file as a data source. By adding the database file as a data source IconAuthor can access it. ODBC drivers provide the connection between ODBC and the individual databases. ODBC drivers are generally available from the database manufacturer or from a third party vendor. Each database package needs its own driver.

IconAuthor provides you with the following ODBC drivers:

Access
Btrieve
dBase
Excel
FoxPro
Paradox

Important: These drivers may not be distributed with your applications. Contact the database manufacturer for information on distributing drivers.

Database Topics:

Database Styles
ODBC Administrator
Creating a Database
Steps for Using the Database Object
Using Bound Objects
Navigating through a Database
Updating and Adding Records

Database Styles

Right-click on a Database object and choose Database Styles... from the pop-up menu to display the choices for the different kinds of database styles. The two available styles are Control Bar and Iconic.

Select the desired style and choose OK to close the dialog box. When you subsequently open the Properties dialog box, you will find that the selection of properties varies depending on the style of database you have chosen.

Iconic Database Style

The default database style is Iconic. It displays as an icon in the SmartObject editor but is invisible at runtime. It requires your application to manipulate the database via IconAuthor icons and SmartObject displays. Because there is no graphical interface for this style, you need to create your own interface for navigating the database using SmartObject Editor objects.

Control Bar Database Style

The Control Bar database style appears both in the Editor and when you run your application. It provides a visual control bar, which is shown in the figure to the right. The control bar lets the user navigate to the First, Previous, Next and Last records and Update the current Record. The Control Bar is scalable and has additional properties for enabling and displaying its individual buttons.

ODBC Administrator

The ODBC Administrator, located in your Control Panel group, is used to register, add and remove data sources available for the Database object. A data source represents any database file that you want to use with the Database object. Typically, you would add each database file individually through the ODBC Administrator. You can also register data sources at runtime by setting the objects Command property to the DataSourceConfig command. This is useful if you want to provide a complete installation for your customers.

Related Topics:

Adding Data Sources

Adding Data Sources

Every database file you create needs to be added as a data source so that IconAuthor can access it via the Database objects ConnectString property. If you do not add your file as a data source, your IconAuthor application will not be able to access it.

Double-click on the ODBC Administrator icon (located in your Control Panel group) to open the Data Sources dialog box.

The Data Sources dialog box lets you add or delete data sources and install ODBC drivers. After you choose an activity, provide information to the dialog box that appears. There is also online help available for every dialog box. To add a data source, use the following instructions as a guideline.

To add data sources:

1. Click on the Add button.
2. Click on the ODBC driver that supports your database file then click OK.

   The appropriate ODBC Setup dialog box appears.

   Data Source Name:
   Enter a descriptive name of the database file, such as Customers or Students. This name will appear in the Data Sources list box.

   Description:
   Enter a description of the database file. This is for your own purposes. IconAuthor cannot access this information.

3. Click on Select Database... to find the database file.

   The Select Database browser appears.
4. Click on the database filename you want to add and click OK.
5. Click OK in the ODBC Setup dialog box.

The database file you added now appears in the Data Sources list box.

Creating a Database

Create a database file using a database software program that supports ODBC. You must have the appropriate ODBC driver installed on your system. Once youve created the database file, you need to add it as a data source via the ODBC Administrator so IconAuthor can access it. Optionally, you can register data sources at runtime in order to provide a complete installation for your customers. Use the DataSourceConfig command for dynamic data source registration. For more information, see the description of the Command property in Chapter 10.

Using the Database Object

There are three basic steps to using the Database object. Follow these steps when you use the object at edit time or runtime. Note that execution of these steps is interchangeable between runtime and edit time.

1. Establishing a Connection
2. Creating a Recordset
3. Displaying a Recordset

Establishing a Connection

The first step in using the Database object is to establish a database connection. You can set up the connection within the SmartObject Editor or you can use icons to establish the connection at runtime.

To set up the connection in the SmartObject Editor, use the Database objects ConnectString property. Note that the connection wont actually be in effect until you run your application. Set the ConnectString property to an existing Data Source. In the Database Properties dialog box, select the ConnectString property and open the drop-down list to display the installed data sources. If your database file doesnt appear in this list, you have to install it as a data source using the ODBC Administrator.

You could also set the CommandOnCreation property to DataSourceConnect. This automatically connects to the database when the object is created. This is useful if you want to connect to a database without creating a recordset.

To establish a connection at runtime use several icons. First, use an ObjGet icon to retrieve the available data sources. The ObjGet icon retrieves the current setting of the DataSources property. Second, use an ObjSet icon to set the ConnectString property to the appropriate datasource. Then, to establish the connection, use an ObjSet on the Command property using the value DataSourceConnect.

You may have multiple Database objects connected to the same data source. Having many connections open at once may strain your systems resources therefore disconnect any connections when they are no longer needed. To disconnect a connection, use an ObjSet icon on the Database object setting the Command property to DataSourceDisconnect.

Creating a Recordset

Once youve set up a connection, you can then define a recordset using SQL. A recordset represents a set of records selected from a datasource. To define the recordset, use the SQLText property to enter a SQL command. For example, the following SQL command:

Select name, phone number from Customers

will cause the Database object to extract the data from the Name and Phone Number fields of the table named Customers. SQL is a powerful and complex command language. The more SQL you know, the better you can manipulate the Database object. Refer to your SQL documentation for information on SQL usage and syntax.

To create the recordset within the SmartObject Editor, set the CommandOnCreation property to RecordsetOpen. Doing this not only opens the recordset but also implicitly connects to the data source you set in the ConnectString property. This will open the defined recordset which retrieves the first record when the SmartObject page containing the Database object is displayed. The user will then be able to scroll through the records and add, delete and update records, provided that youve given these options.

To define the Recordset at runtime, use an ObjSet icon to set the SQLText property to an appropriate SQL command. Then use another ObjSet icon to set the Command property to RecordsetOpen. This will open the recordset and retrieve the first record when the SmartObject page is displayed.

Displaying the Recordset Data

There are two ways to display the recordset data: using Bound objects or using IconAuthor icons. The quickest and easiest way to display recordset data is to set up your SmartObject page with Bound objects. Then when you display the SmartObject page the first record in the recordset will automatically display in the bound objects. The other way to display recordset data is via IconAuthor icons. This requires greater manipulation of your supporting applications structure.

Once a record displays the user can navigate through the recordset in one of two basic ways, depending on whether your application uses an Iconic or Control Bar style Database object.

Using Bound Objects

CheckBoxes, Radio Buttons, List Boxes, Combo Boxes and Text objects can be set as bound objects. This means you can give each object a specific database field to display. For example, a Text object can be set to display a customer name while a Check Box can be set to indicate whether the customer made a purchase in the last 90 days. As the user navigates through the recordset, the bound objects will automatically be updated with the current records data.

For any bound object, set the DataObjectName property to the ObjectName of the Database object. You also must set the DataFieldName property to the name of the particular field within the database you want it to display. Use the Field Browser from the drop-down list to access a list of all data sources and their field names. For Radio Buttons and Check Boxes, set the DataValueChecked property to the field value it must match in order for it to be filled in or checked. IconAuthor compares the field value with the DataValueChecked property value and acts accordingly. In addition to this for CheckBoxes, set the DataValueUnChecked property to the field value it must match in order to be unchecked.

Use a Display icon to display the SmartObject page. When the SmartObject page displays, the first record in the recordset will automatically display in the bound objects.

Using IconAuthor Icons

Use an ObjGet icon on the RecordData property and enter a variable name in the Variable Name field. The data will be stored in the variable you named in the Variable Name field as a semicolon delimited list which contains all of the data pertinent to the current record. You could also set the data to a variable array. In the Variable Name field, type in the variable name followed by an open and closed bracket. For example, to store the record data in an array called @Result, type in @Result[] in the Variable Name field. If you only want the data from the third field in the record, type @Result[3]. If you want the data from the third and fourth fields, type @Result[3], @Result[4]. This information can then be displayed to the screen or manipulated in other ways via ObjSet icons.

Navigating through a Database

Once a record displays the user can navigate through the recordset in one of two basic ways, depending on whether your application uses an Iconic or Control Bar style Database object.

Navigating Through the Recordset Using Icons

Once youve displayed the database file, you can navigate through the recordset by using ObjSet icons to set the Command property of the Database object.

The Command arguments are:

RecordSeekFirst	RecordSeekPrev
RecordSeekLast	RecordSeekTo
RecordSeekNext	

In order to use the RecordSeekTo command, you must set the PositionSeek property to a record number. To find the number of records in the recordset, an ObjGet on the RecordCount property can be performed.

You can also set up Push Button or Picture Push Buttons to let the user navigate through the recordset. For these two objects, set the ControlObjectName to the ObjectName of the database object. Then set the ControlCommand property to one of the command arguments listed above. The user will then be able to click on a Push Button or Picture Push Button to navigate through the database.

Navigating Through the Recordset Using the Control Bar

The default for the AutoNavigate property is True. This lets the user click on any button to navigate through the recordset. Events can be enabled for the clicking of individual buttons by setting the NotifyOnRecord- properties to True. If bound objects are associated with the recordset, any updates to the fields will be performed automatically.

If you set AutoNavigate to False, you can evaluate any events the user may have generated by clicking on the control bar buttons and then execute the appropriate icons. For example, if the user changed a value in the record and then clicked the Update button, you could evaluate the change before letting the Update occur. When AutoNavigate is set to True, all Updates happen automatically.

Updating and Adding Records

You can use an ObjSet icon to set the RecordData property to update the current record or you can change the contents of a bound object and then navigate to a different record. To change record data via Text bound objects, you must set the Editable property of the object to True. With the Control Bar style, the user can change the contents of the record and press the Update button. Any changes will be updated automatically.

For additions, the current record must be the special EOF (End of Field) record. This record can be positioned to only by setting the PositionSeek property to EOF and then doing the RecordSeekTo command. Once positioned on the EOF record, any subsequent ObjectSets on the RecordData property will add new records to the database.

Related Topic:

Record Locking

Record Locking

A record in a recordset can be updated by an explicit ObjectSet of the RecordData property or by a navigation command if bound controls are tied to the recordset. There are two common modes of record locking; optimistic and pessimistic. The Database object supports optimistic record locking. Optimistic record locking locks the record only for the duration of an update. When using optimistic record locking in a multi-user environment, the application must handle an update failure. Pessimistic record locking locks the record as soon as the record is made the current record. Pessimistic locking has a performance penalty since concurrent access to the same record is prohibited.

Database Object Properties

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

Important: Several of these properties are not in the editor but are available at runtime via ObjGet and/or ObjSet icons in IconAuthor.

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. When you open the Properties dialog box, you will find that the selection of properties varies depending on whether the object is live or static.

Related Topics:

Making a Graphic Object Live or Static
Loading a Graphic into a Graphic Object
Graphic Object Appearance
Graphic Object Interactivity
Graphic Object Properties

Making an Object Live or Static

Graphic, Text and OLE objects are live by default. To make an object static right click on it to display the pop-up menu and choose Object Type... Select Static and choose OK to close the dialog box. If you decide to change the object back to live, you can re-open the ObjectType dialog box at any time.

Loading a Graphic into a Graphic Object

A key property of a Graphic object is called FileName. You set FileName to the name of the file you want to appear in the object. Graphic objects give you a choice of linking or embedding a file. When you embed a file it achieves the same result as pasting the image into the object. The embedded image becomes part of the SmartObject file. When you link a file, it remains separate from the SmartObject file. At runtime (if the object is live), you can use an ObjSet icon to change the FileName property of the object to a different file, thereby linking a new file to the object. This allows you to use one Graphic object to show different graphics at runtime.

You can also paste an image (previously cut or copied to the Clipboard) into a Graphic object. Cut or copy an image to the Clipboard. Right click on a Graphic object and choose Paste. Pasting an image from the Clipboard is the same as embedding a file. The image becomes part of the SmartObject file.

Graphic Object Appearance

By default, if you draw a Graphic object and then set its FileName property, the object is resized according to the size of the graphic file. This feature is controlled through the DrawStyle property. You can also use this property to scale, clip or tile the graphic within the object. If desired, you can also display a border surrounding a graphic. This characteristic is controlled by the Border property which is True by default.

The ColorTransparent property lets you specify a color, used in the graphic, that you want to appear and behave transparently at runtime. As an 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. Any transparent part of the graphic is no longer considered an active part of the object.

Graphic Object Interactivity

A user can interact with a Live Graphic Object in several ways.

Live Graphic objects can behave like Push Buttons.
Live Graphic objects can behave as touch screen buttons.
Live Graphic objects can be cursor sensitive.
Live Graphic objects can be dragged.
Live Graphic objects can be drop targets.
Static Graphic objects can be selectable hotspots.

Objects as Touch Screen Buttons

Use the NotifyOnPressLeft property to allow users to interact with touch screens. When the property is True and the user touches 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.

Objects as Press and Hold Areas

You can also use the NotifyOnPressLeft property to generate an event as soon as the user presses the left mouse button on an object. As an example, this can be useful if you want to let the user press on an object to show information about the object in a status bar.

Graphic Objects the User Can Drag

Use the Dragable property to make a live Graphic object dragable. When the user moves a dragable object, the DragMode property controls whether the object is simply moved, or copied. Use the DragAction property to control how the user initiates the move or copy operation by specifying Click or Drag.

Related Topics:

How the Object Appears when Dragged
Where the Object can be Dropped
Detecting when the Object is Dropped
Detecting where the Object is Dropped

How an Object Appears when Dragged

The DragCursor property controls whether the cursor is visible when an object is being dragged. If you set this to True the cursor remains visible on top of the object. If you set it to False, the cursor disappears from view. The DragGraphicYes and -No properties control how the object appears as it is dragged over valid or invalid drop positions. The DragGraphicYes property sets the graphic file that appears when the object is in a valid position and the DragGraphicNo property sets the invalid graphic. Optionally, you can use the DragTransparentColor property to make a specific color in the DragGraphicYes and/or -No graphics transparent. This is particularly useful, for example, in creating the effect of a polygon-shaped, dragable object.

Where the Object can be Dropped

Use the DragType property to identify where an object can be dropped. While a dragable object has a DragType value, a target object has a corresponding DropType value. (Only live Graphic objects and Transparent objects have a DropType property and can therefore serve as targets.) An object can only be dropped on a target if the dragable object's DragType value matches the target object's DropType.

The DragBringToTop property sets how the object appears when it is dropped. This property lets you decide whether the object will appear in front or in back of any other live objects on the screen when it is dropped.

Detecting when the Object is Dropped

Live Graphic objects have several -Notify properties that you can set to True if you want the application to be alerted when the object is dropped. If NotifyOnDragDrop is True, a "DragDrop" event is generated when a user drops an object on a valid target. Conversely, NotifyOnDragFail generates a "DragFail" event when the target is invalid. You can also set the NotifyOnDragAbort property to True so that a "DragAbort" event occurs if the user clicks the right mouse button to cancel a drag action.

Detecting where the Object is Dropped

When a successful drop occurs at runtime, the DragTargetName property is automatically set to the name of the target object. When your application detects that a "DragDrop" event has occurred, an ObjGet icon can retrieve the current setting for this property.

Graphic Objects that Serve as Drop Targets

To set up a live Graphic object as a drop target, you set its DropType property to match the DragType property of the dragable object. You can also set the target's DropPosition property to indicate whether you want the dropped object to appear centered on the target object.

Static Objects as Selectable Hotspots

If a Graphic, Text or OLE object is static, you can set its SelectionArea property to designate it as a selectable hotspot at runtime.

Related Topics:

Creating Input Selectable Objects

Graphic Object Properties

The following lists show all of the Graphic object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Click the right mouse button on an IconAnimate object and choose IconAnimate Styles... from the pop-up menu to display the choices for the different kinds of IconAnimate styles.

Select the desired style and choose OK to close the dialog box. When you subsequently open the Properties dialog box, you will find that the selection of properties varies depending on the style of button you have chosen.

Note: The Movie object lets you play third-party animations in your applications.

Related Topics:

Playing an Animation
IconAnimate Object Properties

Playing an Animation

There are two basic ways to play an IconAnimate object. You can use the objects CommandOnCreation and Command properties to play the object. Or, you can use a Button object to let the user play the file.

IconAnimate Object Properties

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

Important: Several of these properties are not in the editor but are available at runtime via ObjGet and/or ObjSet icons in IconAuthor.

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

Related Topics:

Defining Acceptable Keys
Defining Keys that are Also Recognized by other Objects
Detecting When a User Interacts via the Keyboard
Learning Which Key the User Activated
Keyboard Object Properties

Defining Acceptable Keys

Set the KeyboardList property to specify the keys that the user can press. You can type the key names in, using a semi-colon to separate one key or key combination from another, or you can choose an option from the drop-down list. The options are either All or Key Selector... If you choose All, the user will be able to press any key on the keyboard.

If you choose Key Selector..., the Key Selector dialog box appears. The Key Selector lets you specify keys the user can press. To select a key, press the Record button to get in Record mode. While you are in Record mode, you can select as many keys as you want by clicking on them or pressing the appropriate keys on your keyboard. Each key that you click on will be added to the Key Sequence listbox. Click on Clear to erase all your choices. When you are finished, click on Stop.

To specify multi-key combinations for a Keyboard object use any combination of SHIFT, CONTROL, or ALT followed by one of the valid single keys described below. Use a hyphen to separate one key from another if they are part of a multi-key combination. For example ALT-F1 is a valid combination.

The following list identifies the keys that are supported. The key names appear as you should select (or type) them within the Key Selector dialog box. Where applicable, a second expression, in parentheses, identifies how the key is generally labeled on your keyboard.

A - Z	F1 - F12	LEFT (Left Arrow)
0 - 9	Numlock	RIGHT (Right Arrow)
NUMPAD0 - NUMPAD9	SCROLL (ScrollLock)	UP (Up Arrow)
BS (Backspace)	HOME	DOWN (Down Arrow)
DEL (Delete)	INS (Insert)	PAUSE
END	TAB	SNAPSHOT (PrintScreen)
RETURN	NEXT (Page Down)	CAPITAL (CapsLock)
ESC *	PRIOR (Page Up)	

* The ESC key is used in IconAuthor to escape from a running application. This key can be used in Present, as long as it is not defined as the BreakKey.

To specify an international character above ASCII #125, click on Stop. Hold down the ALT key and type the ASCII number on the keypad. The resulting character will be added to the Key Sequence list box.

Once you click on Stop, you are in Edit mode. In Edit mode you can delete an entry by selecting it and pressing DELETE. You can also type in keys and key combinations, separating each key and key combination with a semicolon. Make sure the last entry is not followed by a semicolon.

Click on OK to close the Key Selector dialog box. The keys you selected will automatically be added to the KeyboardList property.

Defining Keys that are Also Recognized by other Objects

In some situations, you may define a key for the Keyboard object that could potentially be recognized by another object as well. For example, your application may set up a Keyboard object to recognize the "A" key. The same display may include an editable Text object (in which the user could potentially type an "A.") You need to specify what you want to happen if the user presses the "A" key while editing in the Text object.

In another example, you may set up a Keyboard object to recognize the RETURN key. The same display could include a Push Button which by nature can also be activated via the RETURN key.

The basic rule for this kind of situation is that the Keyboard object always gets the keystroke or key combination. You use the KeyboardForward property to control whether you want the keyboard information to be "forwarded" to the 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.

Detecting When a User Interacts via the Keyboard

The Keyboard object can be set up to notify the application when the user presses a key and/or when a user releases a pressed key. If you set the NotifyOnKeyDown property to True, when a user presses a key defined in the KeyboardList, a "KeyDown" event is generated. If you set the NotifyOnKeyUp property to True, when a user releases a key, a "KeyUp" event is generated.

Your application can include an ObjEvent icon to await the user's interaction. When the user interacts, the appropriate event ("KeyDown" or "KeyUp") 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 and how the user interacts.

Learning Which Key the User Activated

When a user's action generates a "KeyDown" or "KeyUp" event, the KeyboardKeyPressed 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 KeyPressed property and store it in a variable. The application can then take action based on the user's selection.

Keyboard Object Properties

The following list shows all of the Keyboard object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Related Topics:

List Box Appearance
Entering Data in List Boxes
Retrieving Data from a List Box
List Box Interactivity
How List Boxes Work with Database Objects
List Box Properties

List Box Appearance

You determine the size of the list box when you draw the object. If there are too many items to all be visible at once, the list box has a scroll bar. Use the Font and ColorText properties to control the font and color of characters in the Combo Box. The ColorBackground sets the color of the area behind the text.

By default, no item in the list is selected. You can however, set up a selected default, using the SelectedItemData or SelectedItemNumber properties. For example, if you set SelectedItemNumber to 10, the tenth item in the list will be selected by default.

Entering Data in List Boxes

You can enter data in a List Box in one of three ways:

Use the List Box Data dialog box.
Set the object's ItemList property.

Using the List Box Data Dialog Box

The List Box and Combo Box objects have a dialog box that allows you to easily specify the items that you want to appear in the list. Right click on a List Box object and choose Item List... from the pop-up menu. The List Box Data dialog box appears.

Type an item into the top text box and click on Add to add the item to the list.

Each time you type an item and click on Add the item is appended to the end of the list. If you click on an item in the list and then choose Add, the item you typed in the top text box is added to the list just above the currently selected item. When you want to append to the end of the list again, choose Clear to remove the highlight from the list.

You can also move an item from one position in the list to another. Click on the item you want to move. The item is highlighted. Move the mouse cursor to the location in the list where you the item to move. Click the left mouse button to move the item.

To remove an item click on the item and choose Delete. To remove all items, choose Delete All. If you change your mind choose Cancel. The dialog box is removed without putting any of your changes into effect. When you are finished with the dialog box and want to accept the changes you have made, choose Done. (The items in the dialog box will be sorted alphabetically if you set the List Box's Sort property to True.)

Retrieving Data from a List Box

You can do an ObjGet on the ItemList property to get the entire list or the name of a specific item in a list and store the information in a variable. For example, if you wanted to get the third item in the list which uses the @list array, you would enter @list[3] in the Variable Name field of the ObjGet icon. To get the entire list, enter @list[].

List Box Interactivity

There are two basic ways in which a user can interact with a List Box. Typically, a user generates an event by either making a final selection from List Box or by changing the currently highlighted item. These kinds of actions can generate events that can be evaluated by your application.

You can also set up a List Box so that a user generates an event simply by moving the mouse cursor on to and off of the object. For example, you can set up your application so that as the user moves the cursor over a List Box, a status bar at the bottom of the screen displays the purpose of the object. When the user moves the cursor off the object, the status bar message disappears.

Related Topics:

Making the object cursor sensitive.
Detecting when the User Makes a Final Selection
Detecting When the User Changes the Highlighted Selection
Learning What the User Selected or Highlighted

Detecting when the User Makes a Final Selection

A user can make a final selection in the following ways:

+ Click on an item in the List Box and press Return.
+ Double-click on a list item.
+ Use the arrow keys to change the highlighted list item and press Return.

If you set the NotifyOnSelect property to True, when a user makes a final selection from a List Box a "Select" event is generated. Your application can include an ObjEvent icon to await the user's interaction. When the user makes a final selection, the event "Select" is stored in @_Object_Event and name of the affected object is stored in @_Object_Name. A Branches composite can test these values to detect when a user makes a selection and/or from which List Box the selection was made.

Detecting When the User Changes the Highlighted Selection

A user can change the highlighted selection in a List Box in the following ways:

+ Press arrow keys.
+ Click on a different item.

If you set the NotifyOnSelectChange property to True, when a user changes the highlighted selection in a Combo Box a "SelectChange" event is generated. Your application can include an ObjEvent icon to await the user's interaction. When the user makes a final selection, the event "SelectChange" is stored in @_Object_Event and name of the affected object is stored in @_Object_Name. A Branches composite can test these values to detect when a user changes the highlight and/or in which List Box the action occurred.

Learning What the User Selected or Highlighted

When a user makes a final selection or changes the highlighted selection in a List Box, the object's SelectedItemData property is reset to the user's selection. As an example if the user chooses "blue" from a list of colors, SelectedItemData is set to blue. Once your application detects that a "Select" or "SelectChange" has occurred, an ObjGet icon can retrieve the current setting of the SelectedItemData property and store it in a variable. The application can then take action based on the user's selection.

How List Boxes Work with Database Objects

List Boxes (along with Check Boxes, Radio Buttons, Combo Boxes and Text objects) can be bound to a Database object. When the Database object locates a record, the List Box is automatically set to the appropriate value in the list.

For example, consider a database that contains sales information. The database contains fields such as Salesperson ID and Region. The application uses a Database object to open the database. It also includes a ListBox whose list has four options: Region 1, Region 2, Region 3 and Region 4. The ListBox is bound to the Database object and is set up to display the Region value for the current database record.

As soon as the Database object finds a record, the ListBox is automatically set to Region 1, Region 2, Region 3, or Region 4, whichever is appropriate for the current record. The user can leave the setting as is, or make a different selection from the list. The new selection is updated in the database as soon as another record becomes current.

Note: If you make an error in setting up the List Box and fail to include a potential value (such as Region 3 in the preceding example) the List Box will not be able to select the correct value.

To bind a List Box to a Database object, you need to set the DataFieldName property to the name of the database field you want it to display. You also need to set the DataObjectName to the name you gave the Database object.

List Box Properties

The following list shows all of the List Box object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Select the desired style and choose OK to close the dialog box. When you subsequently open the Properties dialog box, you will find that the selection of properties varies depending on the style of menu you have chosen. An application can only show one of each style of menu at a time.

Related Topics:

Basic Menu Object Functionality
Designing the Menu
Menu Object Interactivity
Menu Object Properties
Menu Item Properties

Basic Menu Object Functionality

The Menu object is different from other objects in several important ways.

+ Each Menu object has an ObjectName and a small list of properties that let you control key functionality such as whether the object is visible.
+ Each Menu Item (menu heading or command) within a Menu object also functions as a discrete object. Menu Items have their own properties that are separate from the properties associated with the overall object.
+ For example, one Menu Item (the Cut command) may be enabled while another (the Copy command) is not. Or, one Menu Item may have an accelerator key combination while another does not.

Designing the Menu

Use the Menu object's Menu Design dialog box to create a menu. Click the right mouse button on a Menu object and choose Menu Design... from the pop-up menu to display the Menu Design dialog box.

The area at the top of the dialog box lets you set properties for each Menu Item object that you want to be a part of the overall Menu object. The list box and controls at the bottom of the dialog box let you map the layout of the Menu Items. Every time you use the Property fields at the top of the dialog box to define a new Menu Item, the item is added to the list box below. After you have added all the Menu Items, you can use the control buttons in the bottom right corner to set the menu hierarchy (for example, which items are menu headings and which items are commands).

Related Topics:

Entering Menu Items
Formatting the Menu Layout

Entering Menu Items

To enter and set properties for a Menu Item:

1. In the Object Name text box, type the name of the object.
2. Press Return to accept the Object Name.
3. In the Caption text box, type the description of how you want the Menu Item to appear in the menu at runtime.

   You can type a simple name, such as "File." Or you can add special characters to indicate formatting that you want to occur.
4. Press Return to accept the Caption.
5. Optionally, enter a Family Name for the Menu Item.
6. Optionally, set the remaining properties.
7. In the list box, click just below the current line, to move the highlight to the next blank line.

   This highlighted blank line is where the next Menu Item will appear.
8. Click in the Object Name text box and set properties for the next Menu Item.
9. Continue to enter all the Menu Items contained in the Menu object.

Formatting the Menu Layout

By default, after you enter Menu Items in the the Menu Design dialog box, the bottom list box shows the Object Names on the left and the Captions on the right.

Use the arrow buttons to shift the captions so that they depict the hierarchy of the menu. Captions that are flush left are on the same, uppermost level of the hierarchy. To indicate that a command resides in a menu, shift the command caption one position to the right.

To shift a caption to the right, select it and click on the Right arrow button. This moves the item one position to the right. Use the other arrow buttons to move a caption in another direction. This is useful to correct an error if you accidentally move a caption too far to the right, or decide to move a caption to a position under a different menu heading. You can create up to seven hierarchical levels.

In a pop-up menu, if you create two hierarchical levels, the first level items are pull-right menu headings and the second level items are the commands.

Related Topic:

Previewing the Menu Layout

Previewing the Menu Layout

Use the Preview button to view how the menu layout will appear at runtime. When you click on the Preview button, the menu appears below the list box.

You can click on the menu headings and examine the actual commands. After previewing the menu, edit it as necessary. To change an existing Menu Item, click on the appropriate caption in the list box. The property information at the top of the dialog box changes to reflect the currently selected item. Edit the information as desired.

To insert a Menu Item, click on the caption below the desired insertion point, and click on the Insert button. A blank line is inserted in the hierarchy. Set the properties at the top of the dialog box to fill in the blank entry. To delete a Menu Item, click on the desired caption in the list box and click on the Delete button.

When you are satisfied with the Menu Items and layout, click on the OK button.

Special Characters

"&" symbol - Preceding a character with the "&" symbol causes that character to appear underlined. Following Windows 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.

Example: &File generates "File".

"\t" - These characters generate a Tab within the caption. A Tab commonly separates the name of a command from an accelerator key combination. (An accelerator does not automatically work if you specify it as part of the caption. If you specify an accelerator in the caption, you must subsequently define the accelerator using the Accelerator drop-down list.)

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.

Example: \a&Help generates a Help command the is at the far right side of the menu bar.

"-" - The hyphen character causes a horizontal separator to appear in a menu. Specify just the hyphen for the Caption.

Menu Object Interactivity

Neither Menu objects nor the Menu Item objects within them have Notify- properties. This is different from objects such as Push Buttons which require that you specify exactly how a user can interact (left click, right click, etc.) with the object.

You control when a user can choose a Menu Item by 1) setting the entire Menu object to Visible (True) and by 2) setting individual Menu Item objects as Visible and Enabled (True). Once a Menu is visible and the Menu Item is visible and enabled, the user can choose the item by clicking or using accelerator keys.

You also control which actions precede the appearance of a Menu object. This is particularly significant for a pop-up menu which by nature, alternates between non-visible and visible states. Whereas by convention, a Top Level menu is almost always visible at the top of a window, a pop-up menu generally only appears when a user performs a particular action.

Related Topics:

How Top Level Menus Appear and Disappear
How Pop-up Menus Appear and Disappear
How a User Makes a Menu Selection
Detecting which Menu Item a User Selected
Checking and Unchecking Menu Items
Controlling the Pop-up Menu Position

How Top Level Menus Appear and Disappear

Typically, a top level menu is always present at the top of the application window. By setting the overall Menu object to Visible (True) when you create it in the SmartObject Editor, you ensure that the menu bar appears as soon as the SmartObject file displays.

As necessary, the application can hide and re-show the menu bar. For example, the SmartObject page can provide a button (or other interactive means) that the user can manipulate to indicate that they want to hide the menu bar. As soon as the user chooses to hide the menu, an ObjSet icon can reset the overall Menu object's Visible property to False.

It is also possible to remove a menu heading and its menu items from view. Within the SmartObject Editor, assign a menu heading and its component items the same FamilyName. For example, set the Edit menu heading and the Cut, Copy, and Paste commands all to the FamilyName "EditMenu." At various points during execution, you can remove that portion of the menu by setting the family's Visible property to False.

How Pop-up Menus Appear and Disappear

Unlike a top level menu, by convention, a pop-up menu is not initially visible. This means that in the SmartObject Editor, the overall Menu object should have its Visible property set to False (this is the default). In order to make a pop-up menu "pop-up," at runtime you need to use an ObjSet icon to reset the overall Menu object's Visible property to True.

Your structure should include the icons that let the user choose to show the menu. For example, you may want the pop-up to appear when the user right clicks on the window background. In this case, the Window object's NotifyOnClickRight property would be set to True and your structure would evaluate events to detect when the user right clicks on the Window object. When the application detects that this has occurred, an ObjSet icon sets the pop-up menu's Visible property to True. It is important to note that although it is your responsibility to make the pop-up menu Visible at the appropriate point in the structure, once a user selects a Menu Item, the pop-up menu is automatically removed from view.

Note: Once a pop-up menu appears, no other icons execute until the menu is closed.

How a User Makes a Menu Selection

A user always makes a selection from a top level menu by clicking the left mouse button on an item or by using accelerator keys. Once the user chooses an item, the menu items automatically retract into the menu bar.

For a pop-up menu, you decide which mouse button the user employs by setting the MouseButton property to Left or Right. 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. As with a top level menu, once the user chooses an item, the menu automatically disappears.

Detecting which Menu Item a User Selected

When a user makes a selection from a menu (top level or pop-up), an event called "Select" is generated. Your structure should contain an ObjEvent icon to wait for the user to interact. Upon user interaction, the ObjEvent icon stores the event name "Select" in @_Object_Event, and it stores the object name (of the Menu Item) in @_Object_Name. A Branches composite can then evaluate which object the user selected. Once your application detects which Menu Item was selected, the appropriate icons execute the chosen command.

Checking and Unchecking Menu Items

As with any object, it is the responsibility of the author to build the structure that supports a Menu object. If you want an item to be checked or unchecked after a user selects it, you must use an ObjSet icon to set the Menu Item's Checked property. The structure must also include the icons that carry out the actions associated with a checked (or unchecked) item.

Controlling the Pop-up Menu Position

You can set up a pop-up Menu object's display location so that it appears either where the user clicks, or in a designated position. This feature is controlled by the AutoTrack property. When AutoTrack is True, the pop-up menu follows the mouse click. When it is False, the pop-up menu appears where you positioned the Menu object on the SmartObject page.

If AutoTrack is set to True, you can use the Alignment property to control where the pop-up menu appears relative to the cursor position. The Alignment property can be set to Left, Right or Center. As an example, if you set this property to Right, when the pop-up menu appears, the cursor is on its right.

Menu Object Properties

The following lists show all of the Menu object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

Accelerator
Caption
Checked
Enabled
FamilyName
MenuItemCount
MenuItemList
ObjectData
ObjectName
PageName
Style
Visible

Menu objects that are floating pop-up style also use:

Alignment
AutoTrack
MouseButton

Menu Item Object Properties

The following list shows all of the Menu Item properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Related Topics:

Selecting a Movie
Movie Object Interactivity
Movie Object Properties

Selecting a Movie

Use the Filename property to specify the file you want to play in the object. As soon as you load the file, the word "Movie" (in the center of the object) is replaced by the filename. (The filename does not appear at runtime.) Also, by default, the Movie object automatically resizes itself to the appropriate dimensions for the specified file. Set the ResizeToFile property to True to prevent the object from automatically resizing. If you do this, it is possible that some of the image may not be visible if the object is not large enough.

By default the Movie object plays a file from the first to last frame. Optionally, set the PositionStart and/or PositionEnd properties to specific frame numbers if you want to play only a portion of a file.

Movie Object Interactivity

There are three basic ways to use a Movie object.

Providing the User with a Control Bar

To allow the user to control how the video plays, set the ControlBar property to True. This causes a control bar to appear at the bottom of the object. The controls also allow you to preview the movie within the SmartObject Editor. The right-facing arrow button alternately plays and pauses the movie. The square button stops the movie. The scroll bar lets you move to a different point in the movie.

If you provide a control bar, your application does not require icons to support the playing of the movie. As soon as the object displays the user can click on the controls.

The movie plays until it reaches the end of the file (or specified PositionEnd frame), until the object is deleted, or until the user stops or pauses play. You can also setthe CommandOnCreation property to send the object a command as soon as it is created (not waiting for the user to click). As an example, you can use the CommandOnCreation property to send an Open command to pre-load the file for faster play. Or set it to Play so that the movie plays immediately when the object is displayed.

Playing the Movie without a Control Bar

There are two ways to play a movie without the use of a Control Bar. You can use the objects CommandOnCreation and Command properties to play the object. Or, you can use a Button object to let the user play the file.

When you use the Command property, a runtime property called PositionCurrent allows you to retrieve the current frame number. The PositionSeek property lets you specify a frame that you want to seek to at runtime. Once you set the PositionSeek property, you can set the Command property to carry out the seek operation.

All Movie objects have a NotifyonComplete property which (when set to True) generates an event when the file finishes playing. Movie objects also have a NotifyOnError property which generates an event when an MCI error occurs. The resulting error is the current setting of the ResultString property.

Movie Object Properties

The following list shows all of the Movie object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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. For example, you can draw an OLE object, and through it you can access Microsoft Graph to create a chart. When you exit the server application (Microsoft Graph) the chart becomes an object on the SmartObject page. Once an OLE object is part of a SmartObject page, you can double-click on it to access the server once again. (See the Convert To Static command below for an exception to this rule.)

Once you use a server to create data for an OLE object, you decide the action that is used by that particular object at runtime. This is the action that the object takes when a user double-clicks on it at runtime. One option is to use the action that is the default as designated by the server. For example, the server-designated action for Microsoft Draw graphic data is *edit*. The edit action means that when a user double-clicks on the object at runtime, Draw appears and the user can edit the graphic. As another example, the server-designated action for an audio wave file is *play*. When a user double-clicks on the object at runtime, the audio wave file plays.

In order for the play or edit actions to work, the server application must be present. The SmartObject Editor lets you set a property called DefaultAction that specifies whether the object should use the server-designated action, an alternative action, or whether the object should do nothing when the user double-clicks on it. (This last option is obviously effective for an OLE object with data such as a chart or a graphic, but not for data such as a wave file.)

Related Topics:

Making an OLE Object Live or Static
Creating Data for an OLE Object
Permanently Breaking the Server Connection
Pasting Linked Data from the Clipboard
OLE Object Appearance
OLE Object Interactivity
OLE Object Properties

Creating Data for an OLE Object

When you right click on an OLE object and choose Insert New Object... the Insert New Object dialog box appears.

The Object Type list box shows the registered OLE servers available on your system. When you choose a server and click on OK, the application runs and you can create the data (for example a spreadsheet or a sound file). Within the server, choose Update from the File menu before you exit. Then choose Exit & Return to IA Object #xxxx from the File menu. The data you created is displayed in the OLE object on the page.

Permanently Breaking the Server Connection

Once you return data to the OLE object, you have the option of permanently breaking the connection to the server application. When you right click on the object and choose Convert To Static from the pop-up menu, the data within the object becomes static. When the data is static it cannot be changed because the connection to the server has been permanently severed. Static data is different from a static object (an object that is not live). An OLE object with static data can be live and still have its properties, such as Visible and Enabled, changed at runtime.

Pasting Data from the Clipboard

When you right click on the object and choose Paste, you paste data (such as a picture) that was previously cut or copied to the Clipboard (from a server application) into your OLE object.

Pasting Linked Data from the Clipboard

It is also possible to paste data from the Clipboard and maintain the link to the server application. When you right click on the object and choose Paste Link, the data is still linked to the file from which it was originally copied. As a result, any time that you open the server application and change and save that file, the changes you make are automatically updated in the OLE object on the SmartObject page.

Note: The availability of the Paste Link command depends on the application from which the data was copied. If the server application is not DDE-aware, the Paste Link command is not available.

Once you have pasted linked data into an object, you can use the Links... command to review the current status of the link. Choose Links... from the object's pop-up menu to display the Links dialog box. It lets you view information such as the server name and filename. The dialog box also lets you edit the link information. Refer to your Microsoft Windows documentation for more information on changing link information.

OLE Object Appearance

At runtime, when the object's Visible property is True, the appearance of the object varies depending, on the type of data it contains. The DrawStyle property controls the size of the object. The TransparentBackground property 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.

OLE Object Interactivity

OLE Objects have several properties that allow you to control how the object performs at runtime.

What Kinds of Actions are Available

At runtime an OLE object has two possible states, idle and executing. When an object is executing, the action that occurs depends on the current setting of the DefaultAction property. 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 available actions depending on the type of data it contains. Within the SmartObject Editor, you can right click on the object to learn about its associated actions. For example, if you right click on an OLE Sound object, the bottom option in the menu is a pull-right menu heading called Sound Object. If you open the pull-right menu it displays the Edit and Play options. (Note that if an object only has one available action, such as a Word Document which can only be edited, no pull-right menu appears.)

Letting the User Manipulate the Object Directly

If the object's Visible property is True, and the DefaultAction property is set to an action (other than none) the user can manipulate the object directly by double-clicking on it. For example if the user double-clicks on a Microsoft Word object, Microsoft Word appears and the user can edit the existing data (document). Using an ObjSet icon you can change the DefaultAction property of an OLE object at any time. For example, a Sound object that is initially set to play, can later be set to edit.

OLE objects also have NotifyonStart and NotifyOnComplete properties which when set to True, generate an event when the OLE object begins or completes an action, respectively. For example, if you want the application detect when the user double-clicks on a Sound object to play it, set NotifyOnStart to True. Conversely, if you want the application to detect when the sound data finishes playing, set NotifyOnComplete to True.

Activating a Hidden OLE Object

The State property, which is only available at runtime, lets you manipulate an OLE object regardless of whether it is visible. This property can be set to idle or executing. The executing value causes the object to perform its current DefaultAction. For example, consider a page that contains two objects: a Graphic object and an OLE Sound object. The Graphic object is visible and contains an image of a car. The Sound object is not visible but contains wave audio data that sounds like a car horn. Your application can detect when the user left clicks on the car graphic and immediately use an ObjSet icon to set the State property of the OLE object to executing, thereby playing the car horn sound. Remember, using an ObjSet icon you can change the DefaultAction property of an OLE object at any time.

Note: The State property can also be used to manipulate an object that is visible. For example, you can have an OLE object that contains Animation data appear, and immediately use an ObjSet icon to change the State property to executing.

The NotifyonStart and NotifyOnComplete properties of OLE objects are available whether the object is visible or not. If set to True, these properties generate an event when the OLE object begins or completes an action, respectively. For example, if you want the application to detect when Sound data finishes playing, set NotifyOnComplete to True.

OLE Object Properties

The following lista shows all of the OLE object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

When you right click on a Timer object and choose the Styles... command, the Timer Style dialog box appears. The following styles are available:

Periodic
Count Down
Count Up
Alarm

To understand Timers you must remember the concept of events. An event is an action (performed by a live object) that is recognized by your application. For example, when a user clicks on a button or makes a selection from a list box, that is an event. When a Timer count down reaches 0, that is also an event.

When an event occurs the name of the object that generated the event is stored in the system variable @_Object_Name. At the same time, the name of the event is stored in the system variable @_Object_Event. Specifically, when a Timer object event occurs, the string "alarm" is stored in @_Object_Event.

You can set up your application so that when a Timer event occurs, execution flows in a particular direction based on the event. For example, your application can include a Timer that counts down from 2 minutes to 0 each time a test question is given to a user. When the Timer reaches 0, an event occurs, causing your application to display a message to the user that the time allotted for the question has expired. Although a timer object appears as a clock in the SmartObject Editor, it cannot be seen at runtime. Also, your application cannot have more than ten timers.

Related Topic:

Timer Object Properties

Periodic Timers

A Periodic Timer causes an event to occur every x seconds. You set the period length via the TimerData property. The TimerData should be expressed in hh:mm:ss for example, 2:30:00 would cause an alarm event to occur every 2 and 1/2 hours and 20 would cause an alarm event to occur every 20 seconds. Each time the timer reaches 0 it resets itself.

Count Down Timers

A Count Down Timer gives the user a fixed amount of time to perform a task. The TimerData should be expressed in hh:mm:ss. For example, 5 gives the user 5 seconds, 5:0 gives the user 5 minutes, and 5:0:0 gives the user 5 hours.

Count Up Timers

A Count Up Timer lets you determine how long it took a user to do a task. In this case, you would set the TimerData to start at 0. When an event occurs, such as a user clicking OK to move on to the next question in a quiz, you can use an ObjGet icon to retrieve the TimerData property of the Timer object. The TimerData property will tell you the time in seconds. For example, the value 120 equals 120 seconds or 2 minutes.

Alarm Timers

An Alarm Timer causes an alarm event to occur at a specific time of day. Use the TimerData property to set the time at which you want to the alarm event to occur. Be sure to express the TimerData in military time. If you want a kiosk to start up every day at 8 am, you could set the TimerData to 08:00:00 and set the Style to Alarm.

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 Object Properties

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

Timer Object Properties

The following list shows all of the Timer object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

DeleteProtected
Enabled
FamilyName
ObjectData
ObjectName
PageName
Style
TimerData

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.

Related Topics:

The Transparent Object's Visible Property
Transparent Object Interactivity

The Transparent Object's Visible Property

Like many other objects, the Transparent object has a Visible property. Remember that for Button objects, not only does the Visible property remove the button from view, but it also makes it so the user cannot click on it. The Transparent object has a Visible property so that you can disable it simultaneously with other objects. For example, your application can have a family of objects called ColorButtons. The ColorButtons family is made up of three Push Buttons and a Transparent object. Since they all have the same FamilyName, you can change the Visible property of the ColorButtons family to False and remove from view/disable the entire family at once.

Transparent Object Interactivity

Transparent objects can be interactive in the following ways:

Transparent Objects as Push Buttons
Transparent Objects as Drop Targets
Transparent Objects as Touch Screen Buttons
Transparent Objects as Press and Hold Areas

Transparent Objects as Drop Targets

To set up a Transparent object as a drop target, you set its DropType property to match the DragType property of a dragable (Live Graphic) object. You can also set the target's DropPosition property to indicate whether you want the dropped object to appear centered on the target object.

Transparent Object Properties

The following list shows all of the Transparent object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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.

Related Topics:

Making a Variable Object Live or Static
Entering Variables into a Variable Object
Variable Object Properties

Making a Variable Object Live or Static

If you make a Variable object live it can be changed at runtime via object icons in IconAuthor. If you make it static, it cannot be changed at runtime. When you open the Properties dialog box, you will find that the selection of properties varies depending on whether the object is live or static.

Variable objects are static by default. To make an object live, right-click on it to display the pop-up menu and choose Object Type...

The Object Type dialog box appears as is shown in the figure to the left. Select Live and click OK to close the dialog box. If you decide to change the object back to static, you can re-open the Object Type dialog box at any time.

Entering Variables into a Variable Object

The Variable Window is where you type in your variable data. To access the window, right click on the Variable object and choose Variable Window.

The eight sizer blocks let you make the window as big or as small as you want it. To change the size of the window, left click on one of the resizer blocks, hold down the left mouse button as you drag the window to the size you want then release the left mouse button.

To enter text, double-click in the window. The sizer blocks will disappear and a cursor will appear in the window. You are ready to type in your variable information. Type the variable name on one line, press ENTER and type its assigned value on the following line. Enter as many variables and values as necessary. By default, the WordWrap On option is turned on. If you dont want your text to automatically wrap to the next line, you can turn this option off by right-clicking on the window and clicking on WordWrap On.

Once you have defined all of your variables, you can right click on the window to display the pop-up menu. From here you can click on Properties to display the Properties dialog box. To load your variable information into the Variable object, set the CommandOnCreation property to PutAll.

Variable Object Properties

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. Text objects can be display-only or interactive.

Related Topics:

Making Text Objects Live or Static
Text Object General Appearance
Text Object Interactivity
How Text Objects Work with Database Objects

How Combo Boxes Work with Database Objects

Text objects (along with Combo Boxes, Check Boxes, Radio Buttons, and List Boxes) can be bound to a Database object. When the Database object locates a record, the Text object is automatically set to display the appropriate value.

For example, consider a database that contains sales information. The database contains fields such as Salesperson ID and Region. The application uses a Database object to open the database. It also includes a Text object that is bound to the Database object. The Text object is set up to display the Salesperson ID value for the current database record.

As soon as the Database object finds a record, the Text object is automatically set to the Salesperson ID for the current record. The user can leave the setting as is or edit it (if the Text object is editable). The new entry is updated in the database as soon as another record becomes current.

To bind a Text object to a Database object, you need to set the DataFieldName property to the name of the database field you want it to display. You also need to set the DataObjectName to the name you gave the Database object.

Text Object General Appearance

You control several general characteristics of a Text object's appearance.

Block Style - This is the look of the overall object, for example how its border appears (simple or shadowed) and what colors it uses.

Text Content - If you want text to appear in the object by default, you enter it either by typing or by setting the Filename property to specify a text file. You can use the SmartObject Editor to create the text files for display. If the object contains more text than can be displayed at once you can set the ScrollBarVertical property to True to generate a scroll bar.

Text Formatting - Use dialog boxes and menu options to control the formatting of text. You can set character formatting on a per character basis. There are also menu options for paragraph alignment and line spacing.

Text Object Interactivity

User's can interact with Text objects in the following basic ways:

Let the user edit or input text.
Let the user click on a Text object in Push Button fashion.
Let the user click on a Hotword to cause an action to occur.
Make the Text object cursor-sensitive.
Let the user click on a Static Text object to cause an action to occur.

Letting Users Enter Text

To make a live Text Object capable of accepting user keyboard input, set its Editable property to True. When this property is False, the information in the object is display-only. Control the amount of text a user can enter by setting the InputLimit property. For example, if you only want the user to enter up to 15 characters, set this property to 15. You can also control how the user indicates that he or she is finished typing. If you want to require the user to explicitly press the Return/Enter key to denote completion, set InputTerminationRequired to True. Otherwise, you can set up the object so that completion occurs as soon as the InputLimit is reached.

Control the kind of characters a user can input by selecting an input style for the object. For example, if you want a user to only be able to enter a number that represents a quantity of currency, you set the object's input style to Currency. In most cases, when you select an input style, a number of special properties become available for the object. For example, once you set an object's input style to Currency, the CharacterCurrency property becomes available. This property lets you specify the particular currency symbol you want to use. The default is "$."

Related Topics:

Detecting When a User Inputs Text
Learning What the User Typed

Detecting When a User Inputs Text

Use the NotifyOnComplete property to let your application detect when a user finishes typing in a Text object. If this property is True a "Complete" event can be generated in one of two ways. A user can be typing/editing text and press the Return/Enter key to generate a "Complete" event. Or, if InputTerminationRequired is False and the user types and reaches the input limit, a "Complete" event occurs. Your application should include an ObjEvent icon to await the user's action.

You can also set the NotifyOnInputLimit property to True to detect when the user reaches the input limit. When the user reaches the limit (as set via the InputLimit property) an "InputLimit" event occurs. By setting the InputLimitBeep property to True, you can cause the application to generate a beep when the limit is reached.

Learning What the User Typed

Use the Text property (available at runtime only) to determine what the user typed. When the user enters text, the Text property is set to that input. For example if the user types "Stockholm", Text is set to Stockholm. Your application can use an ObjGet icon to retrieve the current setting for the Text property and store it in a variable.

For example, an application displays a Text object and an ObjEvent icon waits for the user to type the name of a city. The user can type up to 15 characters because the InputLimit property is set to 15. NotifyOnComplete is True and NotifyOnInputLimit is False. If the user types 15 characters, nothing happens. However, when the user presses Return, the ObjEvent icon detects the event, the string "Complete" (for NotifyOnComplete) is placed in @_Object_Event and the name of the affected object is placed in @_Object_Name.

Letting Users Click on Hotwords

The SmartObject Editor lets you designate a character, word, or phrase in a Text object as a Hotword. When you set text as a Hotword you are actually assigning a special text style. By default, the style causes text to appear green and underlined. You can change the default appearance of the Hotword style, or you can create your own Hotword style. For example, your own style could be red instead of green.

If you set a Text object's NotifyOnClickHotword property to True, when a user clicks on a Hotword at runtime, a "ClickHotword" event is generated and the Hotword property is set to the word that was clicked. 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.

Use the CursorNameHotWord property to control how the cursor appears over any hotword. Use the HotWordHighlight property, set to True, to cause a hotword to be reverse highlighted when the cursor is over it. To change the color of a hotword when the user clicks on it, use the HotWordColor property. Use the Solid Color Editor from this propertys drop-down list to choose a color value. This property is available within the SmartObject Editor and within IconAuthor.

Related Topics:

Setting a Hotword
Removing Hotwords
Setting Indexed Hotwords
Creating Custom Hotword Styles
Setting Multiple Hotwords
Working with Hotwords in Formatted Text Files

Setting a Hotword

You use a Text object's pop-up menu to quickly set selected text as a Hotword.

To set a Hotword:

1. Double-click on a Text object.
2. Double-click on a word to select it. (Or drag to select a character or phrase.)
3. Right click on the Text object.
4. Choose Hotword from the pop-up menu.

   The selected word changes color and becomes underlined.

Removing Hotwords

To remove a Hotword setting select the word and choose Hotword from the pop-up menu again. To remove all Hotwords from a Text object, choose Remove Hotwords. When you remove the Hotword designation from text, it takes on the style used by the character immediately to the left. If there is no character to the left, it uses the style to the right.

Setting Indexed Hotwords

Optionally, you can assign a numerical index to a Hotword and the same index can be shared by other 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.

This is particularly useful in cases where you want the same action to occur for multiple Hotwords. You would use indexed Hotwords, for example, if you want one pop-up Text object to appear for "car" and "bus" and another Text object to appear for "canoe" and "rowboat."

To set an indexed Hotword:

1. Double-click on a Text object.
2. Double-click on a word to select it. (Or drag to select a character or phrase.)
3. Right click on the Text object.
4. Choose Hotword Data... from the pop-up menu.

   The Hotword Data dialog box appears. By default, all Hotwords have the index 1.
5. Type the index number you want to use for the Hotword.

   Do not use 0 for an index. The 0 index is reserved for non-Hotword text.
6. From the Style drop-down list, choose the Hotword style you want to use.

   The default Hotword style is always available. (Other styles will appear if you have used the Text Styles dialog box to create your own alternative Hotword styles.)
7. Click on OK.

Creating Custom Hotword Styles

There are two ways to create custom Hotword styles. You can change the style that comes with the SmartObject Editor, called "Hotword." Or, you can create your own style. If you create a style, its name must begin with the word "Hotword," such as "HotwordBlue" or "HotwordMine."

To create or edit Hotword styles, use the Text Styles dialog box. To open the dialog box, right click on text that you are editing and choose Text Styles... from the pop-up menu.

Setting Hotwords with Custom Styles

To apply your own style to selected text, you must use the Hotword Data dialog box.

Setting Multiple Hotwords

The SmartObject Editor provides a special feature for globally setting multiple Hotwords within a Text object. For example, within an object, you can automatically set every instance of "lion," "tiger," and "bear" as a Hotword.

First, you use the SmartObject Editor's ASCII text mode (or any ASCII text editor such as Notepad) to create a file that contains a list of each Hotword you want to set. Name the file with an .HWD extension. Second, you use the Apply Hotwords... command to apply the information in the file to a selected Text object.

Minimally, the .HWD file contains a list of Hotwords. For example, the file that sets "lion," "tiger," and "bear" would appear as follows.

lion
tiger
bear

This would set all instances of these words to an index of 1 and the default Hotword style. As necessary, you can include more information in the file to specify the index you want the Hotword to use, and the Hotword style you want to use. The syntax for a Hotword file specification is: Hotword,Index#,HotwordStyle.

As an example, the following file sets Hotwords, indexes and styles.

vehicle,2,Hotword
bus,3,HotwordBlue
car,3,HotwordBlue
canoe,4,HotwordRed
rowboat,4,HotwordRed

You must list a Hotword. The index and style are optional. Here are some examples:

Specification	Result
canoe	Sets the Hotword "canoe" to the default index 1 and the default Hotword style.
canoe,3	Sets the Hotword "canoe" to index 3, using the default Hotword style.
canoe,,HotwordBlue	Sets the Hotword "canoe" to the default index 1, using the style HotwordBlue.
canoe,3,HotwordRed	Sets the Hotword "canoe" to index 3 using the style HotwordRed.

Related Topics:

Hints for Creating Hotword Files
Applying a Hotword File

Hints for Creating Hotword Files

Hotword file settings are not case sensitive. This means that if you set "king" as a Hotword, when you apply the file to a Text object, both "King" and "king" will become Hotwords.

The information in a Hotword file is processed from the top down. This means for example, that if you list "King,2", "Arthur,2" and "King Arthur,3" the third specification will be the one put into effect if "king" or "arthur" appear together.

Any custom Hotword styles referenced in the file must have been previously created. For example, in the previous table, the style HotwordRed was used. Because this style does not come with the SmartObject Editor, you would have to create it to use it.

Use caution when including spaces in a Hotword file specification. If you include extra spaces before the Hotword, such as " Vanilla" the extra spaces will be ignored when the file is applied. If however, you use trailing spaces as in "Bus ", during application the SmartObject Editor will look for "Bus ".

Applying a Hotword File

Use the Apply Hotwords... command to attach the information in an .HWD file to a Text object.

To apply a Hotword file:

1. Double-click in the Text object.
2. Right click on the object.
3. Choose Apply Hotwords... from the pop-up menu.
   An Open dialog box appears. For detailed information on using this dialog box, see Appendix A.
4. Find the .HWD file you want to apply and choose OK.
   The specifications in the file are automatically applied to the Text object.

Working with Hotwords in Formatted Text Files

The preceding sections describe how to work with Hotwords if you are running the SmartObject Editor in the default Page Layout mode. It is also possible to run the editor in Formatted Text mode and still use the Hotword features. The Text menu contains the Hotword, Hotword Data..., Apply Hotwords... and Remove Hotwords commands.

Text Object Properties

The following lists show all of the Text object properties. Some of these properties do not appear in the object's property dialog box because they can only be manipulated at runtime.

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

Setting an Input Style

The term "input style" refers to text that a user inputs as well as text that the author inputs. When you set the input style you are controlling the kind of characters that can be entered in that text object and in some circumstances, the positioning and quantity of text allowable. As an example, you can set the input style to Numeric so that the user can only enter numbers in that text object.

When you right mouse click on a text object, a pop-up menu appears. Choose Input Styles... to display the Input Styles dialog box.

Select the desired option, choose OK to close the dialog box. When you subsequently open the Properties dialog box, you will find that the selection of properties varies depending on the style you selected.

The following input styles are available.

Standard
Alphabetic
Alphanumeric
Currency
Financial
Fixed Decimal
Logical
Numeric
Picture

Standard

This is the default style. It allows any characters and symbols and uses the TextCase and MultipleLines properties.

Alphabetic

This is almost identical to the Standard style but allows only alphabetical characters and spaces.

Alphanumeric

This is almost identical to the Standard style but allows only alphabetical characters, numeric characters, and spaces.

Currency

Allows numbers only and uses the following properties:

CharacterCurrency
CharacterDecimal
CharacterThousands
DecimalPlaces
Baseline

The following example shows how input might appear in this style:

$320,000.00

Financial

Allows numbers only and uses the following properties:

CharacterDecimal
CharacterThousands
DecimalPlaces
Baseline

This property is almost identical to the Currency style except that it does not use the CharacterCurrency property. The following example shows how input might appear in this style:

320,000.00

Fixed Decimal

Allows numbers only and uses the following properties:

CharacterDecimal
DecimalPlaces
Baseline

This property is almost identical to the Currency style except that it does not use the CharacterCurrency and CharacterThousands properties. The following example shows how input might appear in this style:

320000.00

Logical

Allows only one of two characters. You set the two possible characters via the CharacterFalse and CharacterTrue properties. By default, CharacterFalse is set to "F" and CharacterTrue is "T."

Numeric

Allows numbers only with two exceptions: the first character can be a + or - sign *and* the user can enter a decimal point (just one) at any position in the number. With the Numeric style you can use paragraph alignment to control whether input appears on the screen from left to right (make it a left-aligned paragraph) or right to left (make it a right-aligned paragraph). Note that if you use this style, the input cannot be on multiple lines.

Picture

When you choose this style, you are indicating that you want to create a mask that strictly controls which kinds of characters a user can enter and how many. If you choose this style, the Mask property becomes available.

Working with the Style of a Text Object

The style of a text object is what causes it to have certain characteristics such as its color or the width of its border. There are two basic ways to manipulate the style of a text object. You can work with the style by editing it within the SmartObject Editor, or you can work with it by changing certain specific properties at runtime (using an ObjSet icon).

Every text object has a style. When you begin drawing text objects and they are white-filled with a thin black border it's because they have a style attached to them called Default. Whenever you use the text object tool to draw new text objects they use the Default style.

Related Topics:

Changing Block Style
Creating a New Named Style
Removing a Style

Changing Block Style

To select a block style:

1. Click on a text object with the right mouse button.
2. Choose Block Styles... from the pop-up menu.

   The Block Styles dialog box appears.
3. Choose a style from the Styles list.

   Optionally, click on the Custom button to extend the dialog box and change one of the many characteristics of the object to customize it.
4. Choose OK.

Related Topics:

Creating a New Named Style
Removing a Style

Creating a New Named Style

You can create and save a style that you want to use repeatedly, adding it to the list of frequently used styles in the Block Styles dialog box. From that point you simply choose that style from the list whenever you want to use it.

To create a new style:

1. Use the extended Block Styles dialog box to create a style.
2. Choose Save...

   The Save Style As dialog box appears.
3. Type the name of the style in the Style Name dialog box and choose OK.

   The style name appears in the Styles list box. Each time you add a style it is inserted into the list in alphabetical order.

Note: You can create your own default text object style. Click on an existing style in the Styles list and click on the Set As Default button. On subsequent occasions when you start the SmartObject Editor the new Default style is used automatically.

Removing a Named Style

If you do not plan to use a style you can remove it from the list in the Styles box. Use the remove style feature carefully. When you remove a style it is un-retrievable and must be recreated to be re-included in the list.

To remove a style from the Styles area:

1. Click on the style name to highlight it.
2. Choose Remove.

   The style is removed from the list.

Working with Text in a Text Object

After you draw a text object you can put text in it. Use one of the following methods:

Type and edit text via the keyboard
Create a text file and specify it as the Filename property of the object.

Entering Text

The text objects you create can contain any quantity of text. For example, a text object can be empty, contain a single character, or multiple paragraphs.

To begin entering or editing text double click on any text object. The mouse pointer changes to an I-beam cursor and a flashing vertical caret called the **insertion pointer** appears within the text object. If the text object you double click in does not yet contain text the insertion pointer is automatically placed in the top left corner. If the text object already contains text the insertion pointer is placed within the text at the position where you double clicked.

After you place the insertion pointer in a text object you can begin typing. As you type the characters appear on the screen and the insertion pointer moves to the right. In some situations, a text object may require only a word or phrase such as "Help" or "Exit." In other cases text objects will contain multiple paragraphs of text.

The width of a line of text is defined by the left and right margins of the text object. The amount of text you can put in a text object is unlimited. When the insertion pointer reaches the right margin it "wraps around" or jumps back to the beginning of the next line. Press ENTER at any time to start a new line or skip one or more lines. If you make a mistake you can use one of the editing techniques described in subsequent sections or you can simply press the BACK SPACE key to remove text one character at a time.

Related Topics:

Moving the Insertion Pointer
Scrolling Text
Typing IconAuthor Variables
Editing Text
Editing Paragraph Alignment and Spacing
Changing Fonts
Changing Text Color
Working with Text Styles

Moving the Insertion Pointer

If you decide to go back and add a word or sentence to the middle of a line or paragraph you can move the insertion pointer to the desired position and begin typing. There are two ways to move the insertion pointer. You can use the arrow keys or click the I-beam shaped mouse pointer at the desired position in the text. If you use the mouse technique the insertion pointer is placed wherever you click.

By default, when you first double click in a text object Insert mode is in effect. When you type in Insert mode characters are added to the body of text without affecting any existing characters. The alternative mode is Overstrike. Use Overstrike when you want to type and replace a character for every character that you type. To change to Overstrike mode press the Insert key. The current mode is represented by the characters INS (for Insert) or OVR (for Overstrike) in the far right cell of the status bar.

Scrolling Text

When you reach the bottom of the text object the body of text scrolls upward and the insertion pointer wraps around to the beginning of the next line. Each time the body of text scrolls upward the top line moves out of the window and is no longer visible.

Text that scrolls out of view still exists and can be easily brought back into view by scrolling. Use the up and down arrow keys on the keyboard to scroll up and down. For example, if you want to see the top of the body of text, press the up arrow key. Each time you press the up arrow key, the insertion pointer moves up one line. When the insertion pointer reaches the top line of text that is visible, it stops moving and the text begins to scroll down into view.

Note: If you want a user to be able to scroll through the text in a text object at runtime you must set the object's ScrollBarVertical property to true.

Typing IconAuthor Variables

In addition to conventional text, you can type IconAuthor variable names in a text object. At runtime, your IconAuthor application displays the value stored in the variable rather than the variable name.

Any variable used in a SmartObject file must first be properly defined within the IconAuthor application.

Naming Variables

Variable names can contain up to 19 characters (including the '@' symbol). They can be made up of any alphanumeric characters (A...Z, a...z, 0...9), the underscore character '_' and matching square brackets. Matching square brackets are used to indicate indexed variables. For every occurrence of a '[' there must be a ']'. Variable names cannot include any punctuation.

Rules for Expressing Variables

1. Any non-valid character, for example, a space, or a colon (:), terminates the variable name. The space causes a space to appear on the screen directly following the value; the colon causes a colon to appear on the screen.
2. When you want a displayable character to appear immediately following the variable, use a "\" to delimit where the variable ends. When IconAuthor finds a "\" immediately following a variable name, it signals the end of the variable name and is stripped from the output. If you use a "\" to terminate a variable name when it is not necessary, it is still stripped and no errors result.
3. To use "@" or "\" symbols as literals in the SmartObject file they do not require any special treatment unless they are followed by a valid variable character.
4. If a variable name is valid, but IconAuthor cannot find a value for it, a null character (a space) is displayed.

Editing Text

There are several ways to edit text once you've entered it in a text object. You can drag selected text and drop it elsewhere in a text object or in another text object. You can cut or copy selected text to the Windows Clipboard and then paste it into any text object, even one on a different page. Because the Windows Clipboard is shared by many applications, for example, Windows Notepad, you can take the text that you copy or cut to the Clipboard and paste it into a file created with some other text editor.

Related Topics:

Selecting Text
Dragging Text
Removing Text
Copying Text
Pasting Text

Selecting Text

Many text editing tasks require that you first select the portion of text you want to change. You can select any quantity of consecutively occurring text, for example, a character, a word, or several paragraphs.

To select text:

1. Position the I-beam mouse pointer over the first character in the range you want to select.
2. Press and hold the left mouse button and drag the mouse cursor to the last character in the range you want to select.

   As you drag the mouse cursor, the text you select is highlighted.
3. Release the left mouse button.

   The text is now selected and can be altered.

Hints for Selecting Text

The following are some shortcuts for selecting text:

Double-click on a word to select it.

Double-click immediately after a word and select that word and the next word.

Click to position the insertion pointer, hold down the Shift key, and click at another point further down in the text. This selects all text between the two positions .

Click to position the insertion pointer, hold down the Shift key, and press the arrow right key. This selects one character at a time.

Dragging Text

You can drag selected text from one position to another. This can be done within one text object or from one text object to another.

To drag text:

1. Select the text you want to move.
2. Position the I-beam anywhere over the selected text.
3. Press and hold the left mouse button and drag the selected text to the new location (in the same text object or in another text object).

   While you are moving the mouse the appearance of the cursor varies. Whenever the cursor is over a valid drop position (a text object) it appears as an I-beam with a T below and to the right of it. Whenever the cursor is over an invalid drop position (not over a text object) it appears as a circle with a bar through it.
4. When the cursor is over the proper drop position release the left mouse button.

   The text is moved from the original location to the new location.

Note: To copy the text from one position to another instead of moving it, press and hold the CTRL key before you press and hold the left mouse button. Continue to hold the CTRL key until you have dropped the copied text.

Removing Text

One way to remove text is one character at a time. Each time you press the backspace key one character to the left of the cursor is removed. Each time you press the Delete key one character to the right of the cursor is removed. You can also replace any amount of text by selecting it, and then typing new text. When you begin typing, the selected text is removed.

The SmartObject Editor also lets you use the Cut or Clear text with buttons or commands from the Edit menu.

Cutting Text

When you cut text it is stored on the Windows Clipboard. The next text that you cut or copy to the Clipboard replaces the item that is currently stored there.

To cut text:

1. Select the text to be cut.
2. Click on the Cut button.

   Note: Choosing the Cut command from the Edit menu or pressing SHIFT + DEL is the same as clicking the Cut button.

   The selected text is deleted and stored on the Clipboard.

Clearing Text

When you clear text it is not stored on the Clipboard and it no longer exists on the page. Text acted upon with the Clear command is not recoverable.

To clear text:

1. Select the text to be cleared.
2. Choose the Clear command or press DEL.

   The selected text is deleted.

Copying Text

There are two basic ways to copy selected text. You can use the Copy button. You can also press and hold the Ctrl key while you drag selected text to a new location. When you copy text its appearance on the page is unchanged, but a copy of the item is stored on the Windows Clipboard. The next text that you copy or cut to the Clipboard replaces the item that is currently stored there.

To copy text with the Copy button:

1. Select the text to be copied.
2. Click on the Copy button.

   Note: Choosing the Copy command from the Edit menu or pressing CTRL + INSERT is the same as clicking on the Copy button.

   The selected text is copied and stored on the Clipboard.

To copy text by dragging it:

1. Select the text to be copied.
2. Position the I-beam anywhere over the selected text.
3. Press and hold the CTRL key.
4. Press and hold the left mouse button and drag the selected text the new location (in the same text object or in another text object).

   While you are moving the mouse the appearance of the cursor varies. Whenever the cursor is over a valid drop position (a text object) it appears as an I-beam with a + sign with a T below it.

   Whenever the cursor is over an invalid drop position (not over a text object) it appears as a circle with a bar through it.
5. When the cursor is over the proper drop position release the left mouse button.

   The text is copied to the new location. A copy of the text is also placed on the Windows Clipboard. You can paste the text from the Clipboard as many times as you want.

Note: To move the text from one position to another instead of copying it, *do not* press and hold the CTRL key before you press and hold the left mouse button.

Pasting Text

Use the Paste button to paste a copy of text that is currently stored on the Clipboard into a text object. When you paste text onto a page a copy of the item still exists on the Clipboard.

Note: In order to paste text the last item cut or copied to the Clipboard must have been text. If the Clipboard is empty or contains some other kind of item, you cannot paste text.

To paste text:

1. Position the insertion pointer at the location where you want to insert the text.
2. Click on the Paste button.

   Note: Choosing the Paste command from the Edit menu or pressing SHIFT + INSERT is the same as clicking on the Paste button.

   A copy of the text stored on the Clipboard is pasted into the text object at the point where you placed the insertion pointer. You may paste the same text in other text objects until you do the next cut or copy.

Editing Paragraph Alignmenu and Spacing

By default a paragraph is left-aligned and single-spaced. You can change the alignment and line spacing of a paragraph before you enter text in it, or after.

To change the paragraph style:

1. Position the text insertion pointer in the paragraph you want to change.
2. Right Mouse click anywhere in the text object to display a pop-up menu.
3. Choose one of the available options.

Left	Text is aligned along the left border of the text object.
Center	Text is centered within the text object.
Right	Text is aligned along the right border of the text object.
Single	There is no extra space between successive lines of text.
1 1/2	A space that is half the height of the current line, is drawn between successive lines of text.
Double	A space that is the height of the current line is drawn between successive lines of text.

Changing Fonts

You can change the font of selected text or text that you are about to type.

To change the font:

1. Position the text insertion pointer at the point you want to begin using a new font or select a portion of text.
2. Right mouse click on the text.

   A pop-up menu appears.
3. Choose Fonts... from the menu.

   The Font dialog box appears.
4. Make changes as desired, for example, to the font, font size, and font color.
5. Choose OK.

When you choose OK in the Font dialog box the changes you made are reflected in the text you type, or in the text that is selected.

Changing Text Colors

By default, text is black with a white background (fill color). You can change both the text color and the fill color.

To change colors:

1. Position the text insertion pointer at the point you want to begin using a new color or select a portion of text.
2. Right mouse click on the text.

   A pop-up menu appears.
3. Choose Text Color... or Fill Color... from the menu.

   A Text or Fill Color dialog box appears.
4. Choose a color from the Color drop-down list box and choose OK.

When you choose OK the changes you made are reflected in the text you type, or in the text that is selected. Be aware, however, that when text is selected, its colors appear reverse highlighted. For example, red appears as green. When you de-select the text the correct colors will appear.

Working with Text Styles

When you change the style of text you alter one or more of the following characteristics:

font type
font size
font style (bold, underlined, or italic)
text color
text fill color

For example, text in a text object can use any number of the available font types. You can type a title using one font and then change the font characteristics and continue typing several other paragraphs. You can then select a sentence in the middle of one of the paragraphs and change its font characteristics, leaving the surrounding text unchanged.

To select a text styles dialog box:

1. Make sure you are editing text.

   Either the text insertion pointer must be in a text object or a portion of text must be selected.
2. Right mouse click on the text object.
3. Choose Text Styles...

   The Text Styles dialog box appears.
4. Click on a style to select it.

   Optionally, click on the Custom button to alter the font or fill color of the text.
5. Choose OK.

Related Topics:

Creating a New Named Style
Removing a Named Style

Creating a New Named Style

You can create and save a style that you want to use repeatedly, adding it to the list of frequently used styles in the Text Styles dialog box. From that point you simply choose that style from the list whenever you want to use it.

To create a new style:

1. Open the Text Styles dialog box and click on the Custom>> button.
2. Use the Font dialog box, the Text Color dialog box, and/or the Fill Color dialog box to change the current settings.
2. Choose Save...

   The Save Style As dialog box appears.
3. Type the name of the style in the Style Name dialog box and choose OK.

   The style name appears in the Styles list box. Each time you add a style it is inserted into the list in alphabetical order.

Note: You can create your own default text object style. Click on an existing style in the Styles list and click on the Set As Default button. On subsequent occasions when you start the SmartObject Editor the new Default style is used automatically.

Removing a Named Style

If you do not plan to use a style you can remove it from the list in the Styles box. Use the remove style feature carefully. When you remove a style it is un-retrievable and must be recreated to be re-included in the list.

To remove a style from the Styles area:

1. Click on the style name to highlight it.
2. Choose Remove.

   The style is removed from the list.

Creating Text Files

Text objects are the means by which text appears on your SmartObject page. The simplest way to display text is to type it directly into the text object. However, it is also possible to create text in a separate file and then display the file within the text object at runtime.

Text objects have a property called Filename. If you create a text object and set its Filename property to the name of a text file, the contents of the text file is displayed in the object at runtime. Do not type text in a text object that has a text filename associated with it. The text that you type directly into the object will not appear at runtime.

One way to create text files is to work in a different mode within the SmartObject Editor. Simply by choosing New... from the File menu you can change to Formatted Text or ASCII Text mode and begin creating files. Formatted Text mode means that the characters can be formatted (for example, in terms of color, font and size). Text files created in ASCII Text mode have no character formatting.

Another way to create text files is to use any ASCII text editor (such as Notepad) or any Word Processor that supports .RTF (rich text format) files (such as Microsoft Word for Windows).

Text files can be an exceptionally useful way to include text in a text object.

Using Formatted Text Mode

Use the SmartObject Editor in Formatted Text mode to create character-formatted text files that can be linked or embedded in Text objects via the Filename property.

When you use Formatted Text mode to create a text file you are really creating the contents of a Text object. Appropriately, in terms of entering and editing text, Formatted Text mode follows the basic rules for working with text in Text objects. That is, entering text (including IconAuthor variables), and copying, cutting, and pasting text all work the same as they do for a Text object in Page Layout mode.

Text objects have a property called FileName. If you create a Text object and set its FileName property to the name of a text file, the contents of the text file is displayed in the object at runtime. Text objects also have a property called FttFilename. If you create a file using Formatted Text mode, use this property to set the filename.

Like standard .smt files, formatted text (.ftt) files can contain multiple pages.Use the FttPageName property to set the page name of the .ftt file. Do not type text in a Text object that has a text filename associated with it. The text that you type directly into the object will not appear at runtime.

To change to Formatted Text mode:

1. Choose New... from the File menu.

   The New... dialog box appears.

2. Click on the Formatted Text option and choose OK.

A blank, untitled file appears. Also, the ribbon bar disappears and the SmartObject Editor menu items change. The menus now appear as File, Edit, Page Text, and Help.

Formatted Text Mode - Edit Menu

Cut

Deletes the selected text.

Copy

Copies the selected text onto the Clipboard.

Paste

Pastes text into the file at the current position of the cursor.

Clear

Clears the selected text. Cleared text is permanently removed and is not placed on the Clipboard.

Select All

Selects all text in the file.

Formatted Text Mode - Page Menu

Each .ftt file that you create can contain multiple pages.

New Page...

Lets you create a new page in the current SmartObject Editor file.

Page Maintenance...

Displays the Page Maintenance dialog box, which lets you: create, rename, copy or delete existing pages, or import a page from another SmartObject Editor file.

Page Properties

Displays a cascading menu of commands that let you set certain characteristics of the current page.

Color...

Lets you set the color of the current page. This color is not in effect when the text is displayed in a Text Block. To set the color behind the text, use the Block Styles dialog box by right-clicking on the Text Object.

Page Name...

Lets you name the current page.

Next

Displays the next page in the SmartObject Editor file.

Previous

Displays the previous page in the SmartObject Editor file.

Go To...

Lets you jump to an existing page in the current file.

Formatted Text Mode - Text Menu

Fonts...

Accesses the Font dialog box which allows you to define the font characteristics of the text.

Text Color...

Shows the Text Color dialog box that allows you to select a solid color for text.

Fill Color...

Shows the Fill Color dialog box that allows you to select a color for the background of text.

Text Styles...

Shows the Text Styles dialog box that allows you to select or edit a style, create a new style, or remove a style.

Left

Left-aligns text within the current paragraph.

Center

Centers text within the current paragraph.

Right

Right-aligns text within the current paragraph.

Single

Single spaces text within the current paragraph.

1 1/2

1 1/2 spaces text within the current paragraph.

Double

Double spaces text within the current paragraph.

Hotword

Lets you set a Hotword.

Hotword Data...

Lets you set a Hotword with an index and a custom Hotword text style if available.

Apply Hotwords...

Globally applies Hotword status (including indexing and Hotword text style) to the entire .FTT file.

All Hotwords

Makes all words on the current page Hotwords.

Remove Hotwords

Removes all Hotwords from the current page.

Formatted Text Mode - File Menu

Formatted Text mode lets you create text files with in the .FTT format. You can start completely new files in this mode; or you can open an .RTF file created with any Word Processor or editor that supports .RTF. You can also open an .smt file or a .TXT file, but doing so changes the mode to Page Layout or Text mode respectively.

You can save your files as .FTT files. .FTT, .RTF and .TXT file formats can be linked to or embedded in a Text object via the Filename property. Also, an .RTF file can be opened in any Word Processor or text editor that supports .RTF.

Important: Only a sub-set of .RTF formatting is supported. For example, while color and font formatting is supported, complex formatting such as tables, columns and embedded graphics is not.

New...

Displays the New dialog box which lets you choose to open a new SmartObject file in one of the three modes: Page Layout, Formatted Text, or Text.

Open...

Lets you use an Open dialog box to open an existing file. The file you choose to open can be in Page Layout format (.SMT), in Formatted Text format (.FTT), in ASCII text format (.TXT) or in Rich Text Format (RTF.). Depending on which kind of file you choose to open, the SmartObject Editor comes up in the appropriate mode. If you open an .RTF file the editor stays in Formatted Text mode.

Save

Lets you save the current file.

Save As...

Lets you use a Save As dialog box to save the current file as an .FTT file.

Exit

Exits the SmartObject Editor.

Using ASCII Text Mode

Use the SmartObject Editor in Text mode to create ASCII text files that can be linked or embedded in text objects via the Filename property.

To change to Text mode:

1. Choose New... from the File menu.

   The New... dialog box appears.

2. Click on the Text option and choose OK.

A blank, untitled file appears. Also, the ribbon bar disappears and the SmartObject Editor menu items change. The menus now appear as File Edit, and Help.

Entering and editing text in Text mode is similar to using a text editor such as Windows Notepad. Type to enter text. Use the Backspace and Delete keys to remove characters one at a time. Click at a different point in the body of text if you want to reposition the flashing I-beam insertion pointer.

ASCII Text Mode - Edit Menu

To use the Edit menu commands, first drag across the text you want to edit to select (highlight) it. Once the text is selected you can use any of the Edit commands.

Cut

Deletes the selected text.

Copy

Copies the selected text onto the Clipboard.

Paste

Pastes text into the file at the current position of the cursor.

Clear

Clears the selected text. Cleared text is permanently removed and is not placed on the Clipboard.

Select All

Selects all text in the file.

ASCII Text Mode - File Menu

Text mode lets you create text files with in the ASCII .TXT format. You can create and save files to serve many purposes within IconAuthor. For example, .TXT files for displaying text, .VAR files for loading variables, and .fmt files for formatting new database files. Once created, a .TXT file can be linked to or embedded in a text object via the Filename property. Also, a .TXT file can be opened in any Word Processor or text editor that supports .TXT.

New...

Displays the New dialog box which lets you choose to open a new SmartObject file in one of the three modes: Page Layout, Formatted Text, or Text

Open...

Lets you use an Open dialog box to open an existing file. The file you choose to open can be in Page Layout format (.SMT), in Formatted Text format (.FTT), in ASCII text format (.TXT) or in Rich Text Format (RTF.). Depending on which kind of file you choose to open, the SmartObject Editor comes up in the appropriate mode. If you open an ASCII file the editor stays in Text mode.

Save

Lets you save the current file.

Save As...

Lets you use a Save As dialog box to save the current file as a .TXT file

Exit

Exits the SmartObject Editor.

Displaying Video in the SmartObject Editor

The SmartObject Editor allows you to work in video overlay mode in order to display the current video frame. This helps you simulate how the page will appear in conjunction with video at runtime.

To work in video overlay mode:

- Click on the Video Overlay button in the ribbon bar.

  The current video frame (on your video player) is visible in the SmartObject Editor wherever you have used the transparent color. Frequently the transparent color is black, but it depends on the video overlay board installed on your system. Refer to the documentation for your overlay board to determine which color is transparent.

  Note: Video Overlay mode is only available if you have installed and configured the necessary video setup.

Creating Input Selectable Objects

SmartObject files can serve different purposes within IconAuthor applications. Frequently they are straight text and graphic displays that the user is simply intended to view. However, SmartObject files can also play an important part in creating input selectable objects from which a user can make a selection or choose multiple choice answers to a question.

The first step in creating an input selectable menu is to construct the SmartObject Page. Draw **static** text, graphic, and OLE objects so that they include the selections the user can choose from and the instructions to tell the user how to use the screen.

There are two basic ways to make a display input selectable. You can use IconAuthor's Input or InputMenu icon, or the SmartObject Editor to designate parts of the screen as input selectable.

The method that is recommended depends on the number and variety of menus you plan to display within one loop. If you plan to display a menu where the size, number, and location of the selectable areas *are the same* each time the loop is executed, designate input selectability in the IconAuthor Input or InputMenu icon. If you plan to display a menu where the size, number, and/or location of the selectable areas *varies* each time a loop is executed, designate input selectability in the SmartObject Editor.

Related Topic:

Designating Input Selectability in the SmartObject Editor

Designating Input Selectability in the SmartObject Editor

After you create a SmartObject page with only static objects, make one or more objects, input selectable. If your page contains several objects, you can make all, none, or just some of them input selectable. The objects that you make input selectable must be static.

To make an object input selectable:

1. Click the right mouse button on the object you want to make selectable.
2. Choose Object Type...
3. Click on the Static option and choose OK.
4. Click the right mouse button on the object again.
5. Choose Properties...
6. Change the SelectionArea to a unique number.

   If you leave the Area Number blank or set to 0 the object is not selectable.
7. Choose OK .

Repeat the preceding steps as necessary to designate all of the input selectable areas on the page. Note that a page can not have more than 256 input selectable areas. If you click on an object that you have made input selectable, the area number of that object is displayed in the status bar.

After you have created the SmartObject page and designated input selectable areas, access IconAuthor and build a Menu composite into your structure. When you add content to the Display icon, enter the SmartObject file and page as the filename.

When you add content to the Input Menu icon, you define *which areas on the screen you want to be input selectable.* The information that describes the location of the selectable areas goes in the Selection Areas text box. You already designated the input selectable areas within the SmartObject file, so you don't need to do it again. Enter the variable name @_TEXT_AREAS in the Selection Areas text box.

At runtime, the Display icon displays the SmartObject page on the screen, and automatically places the coordinates of the input selectable areas for that page in @_TEXT_AREAS. The Input Menu icon activates the selectable areas on the screen when it sees @_TEXT_AREAS in the Selection Areas text box. The number of the area the user selects is stored in the system variable @_SELECTION. Later, in the Choices composite, the value in @_SELECTION is evaluated and causes a particular branch to be executed.

Managing SmartObject Pages

The blank background on which you place objects is called a page. Each SmartObject file can contain one or more pages that are actually used as separate displays at runtime. The number of pages required by your file depends on your authoring style. Typically, you group pages together in a file because they are logically related to one another. For example, all the pages in one presentation or pageturning sequence are likely to be in one file, or all the pages associated with one part of an application or subroutine might be in one file.

When you first start the SmartObject Editor a blank page called "Untitled" is displayed. If you plan to include multiple pages in the file you will need to give each page a unique name. You also have the option of specifying a color for the page although by default pages are transparent. When you display a transparent page at runtime, the objects on that page appear to rest directly on top of any previously displayed information.

Related Topics:

Creating New Pages
Renaming Pages
Copying Pages
Deleting Pages
Importing Pages
Changing Pages
Working with Page Color

Creating New Pages

Most SmartObject files have multiple pages.

To create a new page in the current SmartObject file:

1. Click on the Page Maintenance button or choose Page Maintenance... from the Page menu.
2. Choose New.

   The New Page dialog box appears.
3. Enter the name of the new page in the Page Name text box.

   The name you use must be unique within the file and can be up to 24 characters.
4. Choose OK.

   If you specify the name of an existing page a message is displayed requesting a "unique" name. Once you have specified a unique page name the dialog box is removed and the new, blank page is displayed.
5. When you are finished with the Page Maintenance dialog box choose Close.

Note: You can also use the New Page button to display the New Page dialog box.

Renaming Pages

To rename a page in the current SmartObject file:

1. Click on the Page Maintenance button.

   The Page Maintenance dialog box appears.

2. In the Page List area click on the page you want to rename.

   The page you select appears in the work area.

3. Choose Rename.

   The Rename Page dialog box appears.

   The current name is highlighted.

4. Enter the new name in the Page Name text box.

   The new name replaces the old one. The name you use must be unique within the file and can be up to 24 characters.

5. Choose OK.

6. When you are finished with the Page Maintenance dialog box choose Close.

Copying Pages

You can copy a page. This is particularly useful if you want to create another page that looks very similar to an existing page.

To copy a page in the current SmartObject file:

1. Click on the Page Maintenance button.

   The Page Maintenance dialog box appears.
2. In the Page List area click on the page you want to copy.

   The page you select appears in the work area.
3. Choose Copy.

   The Copy Page dialog box appears.

   The current name is highlighted.
4. Enter the name of the new page in the Page Name text box.

   The name you use must be unique within the file and can be up to 24 characters.
5. Choose OK.

   The new page is displayed and appears identical to the original.
6. When you are finished with the Page Maintenance dialog box choose Close.

Deleting Pages

You can delete an entire page.

To delete a page in the current SmartObject file:

1. Click on the Page Maintenance button.

   The Page Maintenance dialog box appears.
2. In the Page List area click on the page you want to delete.

   The page you select appears in the work area.
3. Choose Delete.

   A message appears that asks if you want to delete the selected page. Choose Yes to delete or No to cancel the process.
5. When you are finished with the Page Maintenance dialog box choose Close.

Importing Pages

The SmartObject Editor allows you to import pages from other SmartObject files.

To import pages into the current SmartObject file:

1. Click on the Page Maintenance button.

   The Page Maintenance dialog box appears.

2. Choose Import...

   The Import dialog box appears.

3. Choose Browse... to identify the file from which you want to import pages.

   The Browser appears.

   Select a file and choose OK. The Browser is closed and the Import dialog box returns to view. The Import dialog box now contains the name of the file you have specified and the pages that file contains.

4. In the Pages List area, select the pages you want to import.

   To select one page, click on it.

   To select a range of pages, click on the top-most page in the range. Then, while pressing and holding the Shift key, click on the bottom-most page in the range.

   To select several pages that are not a consecutive range, for example, the first and the last page, click on one page. Then, while pressing and holding the Ctrl key, click on any subsequent pages.

   To select all the pages in the file, click on the Select All button.

5. When you have selected all the pages you want, click on Add Pages.

   The pages are appended to the end of the current SmartObject file.

6. When you are finished with the Import dialog box, choose Close.

7. When you are finished with the Page Maintenance dialog box choose Close.

Changing Pages

There are several ways to display a different page in a SmartObject file.

To display the next page:

Click on the Next Page button.

To display the previous page:

Click on the Previous Page button:

To jump to a page:

1. Click on the Go To Page button.
2. The Go To Page dialog box appears.
3. In the Page List area click on the page you want to jump to.
4. Choose OK.
   The dialog box is closed and the selected page is displayed.

Working with the Page Color

The SmartObject Editor lets you change the background color of any page. For example, you can make the entire background blue, red, or transparent. If you make the screen transparent when the page is displayed in IconAuthor, the objects on the page will appear to rest directly on top of the existing screen display.

If you display a page with a color background the color replaces any previously displayed information except for any visible, live objects.

Note: While you are viewing a page with a transparent background within the SmartObject Editor, the background appears white. Do not confuse this with a white background which is not transparent.

You can change the screen color to one of 48 basic colors or you can define a custom color.

To change the screen color:

1. Click the right mouse button anywhere on the page background (not on an object).

   The Page menu appears.

2. Choose Color...

   The Color dialog box appears.

3. Choose a color from the color palette and choose OK.

When you accept a color selection the screen appears with the new color. If you create and then show a new page in the same SmartObject file the screen in that new page has the current color.

Managing SmartObject Files

Managing your SmartObject files means understanding basic tasks such as saving your work, opening existing files, and opening new files.

Assign any legal DOS filename to a SmartObject file. The filename can be one to eight characters with an optional one to three character extension following a period.

Related Topics:

Saving New Files
Saving Existing Files
Saving and Renaming Files
Starting New Files
Opening Existing Files
Printing Files
Path Information

Saving New SmartObject Files

The first time you save a SmartObject file you also name it. If you do not specify a filename extension, the appropriate extension is appended for you. That is, in Page Layout mode the filename ends in .SMT; in Formatted Text mode it ends in .FTT; and in Text mode it ends in .TXT.

To save a new SmartObject file:

1. Choose Save As... from the File menu.

   The Save As dialog box appears.

2. Make sure the dialog box is pointing to the right drive and directory.

   The current path is listed directly below "Directories." Use the Drives drop-down list box to change to another drive and use the Directories list box to change to another directory. Double click on a closed directory folder to open it. When a folder is open any files in that directory (that match the filter in the File Name text box) are listed (greyed) in the list box below the File Name text box.

3. Make sure the Save File as Type box shows the correct kind of file.

   For example, if you are saving a SmartObject file, this box should be set to SmartObject(*.smt). Click on the down arrow to show the available selection of file types.

4. When the dialog box is pointing to the right directory and the file type is correct, you can specify the name you want to assign to the file.

   Choose a filename from the list box below the File Name box or type a new filename into the File Name box.

5. Click on OK.

When you choose OK the dialog box is removed and the file is named and saved. The name of the SmartObject file appears in the title bar.

Saving Existing SmartObject Files

Periodically, it is a good idea to save changes to a SmartObject file that has been saved previously, but now contains unsaved changes.

To save changes to an existing SmartObject file:

- Choose Save from the File menu or press CTRL + S.

  The SmartObject file is saved automatically.

Saving and Renaming SmartObject Files

It is sometimes useful to save a SmartObject file and rename it at the same time. For example, you might want to use DEMO1.SMT as a template. To do so, make a copy of it under a new name, while keeping DEMO1.SMT intact for future use.

If you save and rename DEMO1.SMT with the name DEMO2.SMT, you now have two identical files with different names. Make as many changes as you like to DEMO2.SMT and you will still have a copy of the original DEMO1.SMT.

To save and rename a SmartObject file:

1. Open the file you want to copy.
2. Choose Save As... from the File menu.

   The Save As dialog box appears.
3. Make sure the dialog box is pointing to the right drive and directory.

   The current path is listed directly below "Directories." Use the Drives drop-down list box to change to another drive and use the Directories list box to change to another directory. Double click on a closed directory folder to open it. When a folder is open any files in that directory (that match the filter in the File Name text box) are listed (greyed) in the list box below the File Name text box.
4. Make sure the Save File as Type box shows the correct kind of file.

   For example, if you are saving a SmartObject file, this box should be set to SmartObject(*.smt). Click on the down arrow to show the available selection of file types.
5. When the dialog box is pointing to the right directory and the file type is correct, you can specify the name you want to assign to the file.

   Choose a filename from the list box below the File Name box or type a new filename into the File Name box.
6. Click on OK.

When you choose OK the dialog box is removed and the file is named and saved. The new name of the SmartObject file appears in the title bar.

Starting New SmartObject Files

When you start the SmartObject Editor the work area is blank. You can begin to create the contents of a new file immediately. You can also create a new SmartObject file even if you already have a SmartObject file currently visible in the work area.

To clear the work area for a new SmartObject file:

1. Choose New... from the File menu.

   A new SmartObject file appears *if* the SmartObject file you were working on did not contain unsaved changes.

   If the SmartObject file contained unsaved changes, a message box appears that asks if you want to save the current file. To proceed without saving changes choose No. To save changes before proceeding, choose Yes and a Save As dialog box appears.

2. Make sure the dialog box is pointing to the right drive and directory.

   The current path is listed directly below "Directories." Use the Drives drop-down list box to change to another drive and use the Directories list box to change to another directory. Double click on a closed directory folder to open it. When a folder is open any files in that directory (that match the filter in the File Name text box) are listed (greyed) in the list box below the File Name text box.

3. Make sure the Save File as Type box shows the correct kind of file.

   For example, if you are saving a SmartObject file, this box should be set to SmartObject(*.smt). Click on the down arrow to show the available selection of file types.

4. When the dialog box is pointing to the right directory and the file type is correct, you can specify the name you want to assign to the file.

   Choose a filename from the list box below the File Name box or type a new filename into the File Name box.

5. Click on OK.

When the file has been saved the new, blank file appears.

Opening Existing SmartObject Files

To open a SmartObject file:

1. Choose Open... from the File menu.

   The Open dialog box appears if a) you just started the SmartObject Editor and the drawing area is still empty, or b) you are working on a SmartObject file that does not contain unsaved changes.

   If the SmartObject file contained unsaved changes, a message box appears that asks if you want to save the current file. To proceed without saving changes choose No. To save changes before proceeding, choose Yes and a Save As dialog box appears.

   When you finish using the Save As dialog box the current file is saved and the process of opening an existing file continues.

2. When the Open dialog box appears, make sure the dialog box is looking in the right place for the file you want.

   The current path is listed directly below "Directories." Use the Drives drop-down list box to change to another drive and use the Directories list box to change to another directory. Double click on a closed directory folder to open it. When a folder is open any files in that directory (that match the filter in the File Name text box) are listed in the list box below the File Name text box.

3. Once the dialog box is looking in the right directory you may have to change the value in the List Files of Type field.

   For example to show all files in the current directory, choose All (*.*) in the list box.

4. When the correct file appears in the list box below the File Name text box, click on the filename and choose OK.

   The SmartObject file is opened.

Printing SmartObject Files

SmartObject files can be printed by the page. The information on the page is printed as it appears on the screen.

To print a SmartObject file:

1. Choose Print... from the File menu.

   The Print dialog box appears.

   Note: The Print command uses the current printer selected during Microsoft Windows installation. The printer driver for your printer must be installed on a hard disk drive in your computer system. Use the CONTROL.EXE program in the Microsoft Windows directory to select the current printer and printer communications port configuration.
2. Click to select the page you want to print or choose Select All to select all pages.
3. Choose OK to print the selected page(s), or choose Cancel to close the dialog box without printing.

Path Information

If you are using a customized directory structure to organize the files you create with IconAuthor, the SmartObject Editor, and other editors, you may have to use the Path File... command in the SmartObject Editor Options menu.

When you access the SmartObject Editor, by default, it assumes you are working with files located in the paths defined in your master IAUTHOR.PTH file. If you are creating a SmartObject file for an application that uses paths other than those in IAUTHOR.PTH, choose Path File... to select an alternative .PTH file for the SmartObject Editor to use. Once you specify an alternative .PTH file, the SmartObject Editor will know where to find and store the SmartObject and graphics files for that application.

Use the Path File... command as often as necessary. For example, you may open the SmartObject Editor and decide to work on a SmartObject file used by PROJ1.IWM. Therefore, you choose Path File... and select PROJ1.PTH. At a later point in time, you decide to work on a SmartObject file used by PROJ2.IWM. The files used by PROJ2.IWM are stored in a different group of subdirectories. Therefore, you choose Path File... again, and select PROJ2.PTH.

Quitting the SmartObject Editor

When you are finished working with the SmartObject Editor you can exit the program. By default the next time you start up the SmartObject Editor it will remember the settings you were using. For example, it will remember if you were running the editor full screen or if you had all the features such as the grid turned on or off. The SmartObject Editor remembers the state of these features because by default, Save Settings On Exit in the Options menu is toggled on. Choose the option if you want to toggle it off.

To quit the SmartObject Editor:

- Choose Exit from the File menu.

  If you choose exit and have not yet saved the file you are working on, a dialog box appears and asks if you want to save the changes to the file.

To exit without saving changes:

- Choose No.

To exit and save changes:

1. Choose Yes.

   If the file you are saving has been saved (named) on a previous occasion, it is saved automatically.

   If the file has not been saved before (it is "untitled"), the Save As dialog box appears.
2. Enter a filename in the File Name text box.
3. Choose OK.

   The file is saved and the SmartObject Editor is closed.

When you open the SmartObject Editor from a Content Editor, work with the editor, and close it, the name of the file you last worked on is returned to the Content Editor text box from which the editor was started.

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

SmartObject Editor Contents

The Contents lists the Help topics available for the SmartObject Editor. Use the scrollbar t see entries that are not currently visible.

Keyboard

SmartObject Editor Keys

Commands

Edit Menu
File Menu
Help Menu
Objects Menu
Options Menu
Page Menu

Procedures

Changing Fonts
Changing Text Colors
Creating Input Selectable Objects
Creating Objects
Displaying Video in the SmartObject Editor
Editing Text
Entering Text
Live Object Maintenance
Making an Object Live or Static
Managing Files
Managing Pages
Objects
Object Alignment
Object Layering
Paragraph Alignment and Spacing
Path Information
Quitting the SmartObject Editor
Selecting and Editing Objects
Selecting Text
Setting Object Properties
Setting Object Tab Stops
Typing IconAuthor Variables
Using ASCII Text Mode
Using Formatted Text Mode
Working with Text in a Text Object
Working with the Style of a Text Object

SmartObject Editor Keys

The following accelerator keys and key combinations are available in the SmartObject Editor:

Menu	Command	Key(s)
File	Save	Ctrl + S
Edit	Cut	Shift + Del
	Copy	Ctrl + Ins
	Paste	Shift + Ins
	Clear	Del
Page	New Page...	Ctrl + W
Object (Order)	Bring To Front	Ctrl + U
	Send To Back	Ctrl + D
Options	Full Screen	Ctrl + F
	Hide Tools	Ctrl + T

Additionally, you can use keys to perform text editing functions within a text object.

Key	Function
arrow keys	The up, down, left, and right arrow keys move the insertion point to a new location.
Home	Moves the insertion point to the beginning of the current line.
End	Moves the insertion point to the end of the current line.
Page Up	Scrolls up in the text block.
Page Down	Scrolls down in the text block.
Delete	Deletes the character to the right of the insertion point.
BackSpace	Deletes the character to the left of the insertion point.

SmartObject Editor Commands

To get help with a command, choose the appropriate menu.

File Menu

New...
Open...
Save
Save As...
Delete
Page Setup...
Print...
Printer Setup...
Exit

Edit Menu

Cut
Copy
Paste
Clear
Copy To...
Paste From...

Page Menu

New Page...
Page Maintenance...
Page Properties
    Color...
    Page Name...
    Effect...
Next
Previous
Go To...

Object Menu

Tools
    Select Object
    Audio
    Button
    Combo Box
    Database
    Graphic
    IconAnimate
    Keyboard
    List Box
    Menu
    Movie
    OLE
    Text
    Timer
    Transparent
    Variable
Object Properties (Menu items vary depending on the selected object. With the exception of Properties, the possible menu items are organized alphabetically as follows.)
    Properties...
    1 1/2
    All Hotwords
    Apply Hotwords...
    Block Styles...
    Button Styles...
    Center
    Combo Box Styles...
    Convert Data to Static...
    Database Styles...
    Double
    Fill Color...
    Fonts...
    Hotword
    Hotword Data...
    IconAnimate Styles...
    Import File...
    Input Styles...
    Insert New Object...
    Item List...
    Left
    Links...
    Menu Data...
    Menu Styles...
    Object
    Object Type...

- Paste
- Paste Link...
- Remove Hotwords
- Right
- Single
- Sound Styles...
- Text Color...
- Text Styles...
- Timer Styles...
- Variable Icon
- Variable Window
- WordWrap On

Alignment
- Left
- Right
- Top
- Bottom
- Center Vertical
- Center Horizontal

Order
- Layers...
- Tab Stops...

Options Menu

Path File...
Full Screen
Hide Tools
Grid Alignment...
Confirm Clear
Overlay Mode
Ribbon Bar
Status Bar
Save Settings On Exit

Help Menu

Index
Keyboard
Commands
Procedures
Using Help
About the SmartObject Editor...

SmartObject Editor Procedures

Objects

Audio
Button
Combo Box
Database
Graphic
IconAnimate
Keyboard
List Box
Menu
Movie
OLE
System
Text
Timer
Transparent
Variable
Window

Working with Objects

Creating Objects
Selecting and Editing Objects
Setting Object Properties
Making an Object Live or Static
Object Layering
Setting Object Tab Stops
Live Object Maintenance
Creating Input Selectable Objects

Putting Text in Text Objects

Entering Text
Scrolling Text
Typing IconAuthor Variables
Editing Text
Using Hotwords
Changing Paragraph Alignment and Spacing
Changing Fonts
Changing Text Colors
Working with Text Styles
Working with the Block Style of a Text Object

Working with Pages

Creating New Pages
Renaming Pages
Copying Pages
Deleting Pages
Importing Pages
Changing Pages
Page Color

Working with Files

Saving New Files
Saving Existing Files

Miscellaneous

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.