Contents

For additional assistance, contact Technical Support.

Menu commands

Step-by-step procedures

TB30DLG.DLL reference

Step-by-step procedures

The topics below provide step-by-step instructions for working with the Dialog editor.

Menu commands

For information about the commands on a menu, choose the name of the menu.

- File
- Edit
- Control
- View
- Help

Menu commands

For information about the commands on a menu, choose the name of the menu.

- **File**
  - New
  - Open
  - Save
  - Save As
  - Assign Template to Book
  - Convert template to viewer
  - Exit (Alt+F4)
- **Edit**
- **Control**
- **View**
- **Help**

Menu commands

For information about the commands on a menu, choose the name of the menu.

- **File**
- **Edit**
  - Undo
  - Cut
  - Copy
  - Paste
  - Delete
  - Clear Dialog
- **Control**
- **View**
- **Help**

Menu commands

For information about the commands on a menu, choose the name of the menu.

- **File**
- **Edit**
- **Control**
  - Properties
  - Group
  - Align Controls
- **View**
- **Help**

Menu commands

For information about the commands on a menu, choose the name of the menu.

- **File**
- **Edit**
- **Control**
- **View**
  - Preview Dialog
  - 3D in Preview
- **Help**

Menu commands

For information about the commands on a menu, choose the name of the menu.

- **File**
- **Edit**
- **Control**
- **View**
- **Help**
  - Quick Help
  - Edit Editor
  - About Dialog

TB30DLG.DLL function reference

These functions allow you to display Windows-style dialog boxes and get and set values for dialog box options.

Technical support contact information

Telephone support

Contact Asymetrix at the telephone numbers listed below for information on telephone support contracts.

Australia/Asia Pacific	(61+3) 5255471
Europe (except France and Germany), Middle East, Africa, Russia	44-923-208-433
UK	0800-716-957 (freephone)
France	05-90-83-19 (freephone)
Germany	01-30-81-27-07 (freephone)
USA and rest of world	206-637-1600

Online services

Asymetrix provides complimentary support via fax, Asymetrix BBS, CompuServe, America Online, and Internet to registered users. Technical support responds to online queries within 48 hours (Monday to Friday).

Technical support fax

- Australia/Asia Pacific (61+3) 5255-482
- Europe 44-923-208-419
- USA 206-454-0672

Asymetrix BBS

- Line 1 (1200-2400 baud/9600 baud, US Robotics HST mode) 206-451-1173
- Line 2 (9600-14,400 baud v.32bis) 206-451-8290

America Online

- Find Asymetrix in the Industry Connection, a subset of the Computing and Software area.

CompuServe

- Windows Third Party Developer A forum, section 1 *go asymetrix* or *go winapa*
- Multimedia Vendors forum, Section 15 *go multiven*
- IBM Ultimedia Tools A forum, Section 5 *go ultiatools*

Internet

- techsup@asymetrix.com
- support@asymetrix.com

About Dialog Boxes

Step-by-step

You can have two kinds of dialog boxes in a ToolBook book:

- A ToolBook dialog box, which is a page of a book that you display using a viewer
- A true Windows dialog box, which you create with the Dialog editor and display with a call to a DLL

The two kinds of dialog boxes look and behave somewhat differently. Before choosing one or the other, consider the pros and cons of each kind.

The Dialog editor, a ToolBook application called dialog.tbk, does two things:

- It greatly simplifies the process of laying out the controls in a dialog box.
- When the layout is the way you want it, it translates your layout into the strings of data, called a template, that you pass to the Windows dialog DLL when you want to display the dialog box. It also creates a handler you can use in your application to call the DLL.

See also:

Creating a dialog box
Displaying a dialog box from within a book
How ToolBook displays dialog boxes
Installing a dialog box in a book

How ToolBook displays dialog boxes produced with the Dialog editor

Step-by-step

Dialog boxes produced with the Dialog editor are not stored in the ToolBook application as ToolBook objects. Instead, when the application needs to display a dialog box, it passes two strings of data to a dynamic link library (DLL) called TB30DLG.DLL. One string (called the template) contains specifications for the dialog box -- its size and shape, the controls it contains, and the appearance of the controls. The other string contains initial values for variables associated with controls in the dialog box. The DLL uses the data to create the dialog box "on the fly."

When the user is through using the dialog box and closes it, the DLL returns a data string that contains the final values for the controls. Your application can inspect this string to find out what the user did in the dialog box.

See also:

Creating a dialog box
Dialog boxes versus viewers
Displaying a dialog box from within a book
Installing the dialog box in your application

Using Windows dialog boxes versus using viewers

Step-by-step

Advantages of using a viewer to display a dialog box:

- Creating dialog boxes with viewers is generally faster and easier than using the Dialog editor application.
- You can use multiple fonts and point sizes for text in a viewer.
- You can use ToolBook features such as hotwords in a viewer.
- Objects in a viewer are easier to control (for example, to layer) and to interact with than objects in a Windows dialog box.
- A viewer is contained in your applications .TBK file; you don't need to ship the TB30DLG.DLL with your application.
- Windows dialog boxes cannot have dialog box menus.

Advantages of using a Windows dialog box:

- Dialog boxes conform exactly to Windows standards in details such as the shape of buttons.
- A few controls, such as the "i" icon used often with dialogs, are not conveniently available to viewers.
- Dialog boxes automatically use three-dimensional controls if you choose this option for your application. To use three-dimensional controls in a viewer-based dialog box, you must draw them individually; if you choose to revert to non-three-dimensional controls, you must re-create them.
- If you are accustomed to using the Dialog editor from earlier versions of ToolBook, you may find it easier to use than creating viewers.
- With the Dialog editor, it's easier to create dialog boxes that match those created with earlier versions of ToolBook.
- Windows dialog boxes can take less space in your book. (However, to use them, you must have the TB30DLG.DLL utility available.)

For detail about viewers, refer to the ToolBook User Manual.

See also:

Creating a dialog box
Displaying a dialog box from within a book
Installing the dialog box in your application

Creating a dialog box

Step-by-step

To create a dialog box (general procedure):

1. Open the DIALOG.TBK book.
   The Dialog editor window appears, along with an empty dialog box frame and the tool palette.
2. Select a control from the tool palette and place it on the dialog box. Use standard Author-level techniques to draw, move, and size the control.
3. Open the Properties dialog box for the control and set the properties and initial values.
4. Repeat steps 2 and 3 for additional controls.
5. On the View menu, choose Preview Dialog to see what the finished dialog box will look like.
6. When the dialog box is the way you want it, save it as a file.
7. Assign the dialog box to an application.

The ToolBook tool palette is never displayed in the Dialog editor, but you can use the Command window in the usual ways. Although you can display and use the Script window, any scripts you define are not part of the new dialog box.

See also:

Aligning controls
Assigning a caption
Dialog box controls
Grouping Controls
How to draw a control
How to set initial values for a control
How to set properties for a control
Menu commands
Positioning the dialog box
Previewing the dialog box
Saving a dialog box definition
Setting properties for the dialog box frame
Setting tab order for controls
Sizing the dialog box
The Dialog Box editor tool bar

Dialog box controls

There are 11 controls available for use on dialog boxes. Seven are standard ToolBook controls; four controls are unique to Windows dialog boxes.

The controls are represented by buttons on the tool palette. (The twelfth button is for the selection tool.)

To learn about the features and uses of a control, its properties, and the initial values you can set for it, click the control in the illustration, and then choose the information you want.

See also:

Creating a dialog box

Dialog frame properties

Selection tool

Use the selection tool to select controls and to move and resize them.

Pushbutton

(Standard ToolBook control.)

Pushbuttons are usually used as OK and Cancel buttons. OK buttons accept entries and close the dialog box; Cancel buttons close the box without accepting any entries. In an interactive dialog box, pushbuttons can also be used to trigger other events, such as saving a file or displaying additional options.

See also:
Displaying a dialog box from within a book
Drawing a pushbutton
Pushbutton properties and initial values

Radio button

(Standard ToolBook control.)

Use a radio button when you want the user to select one and only one item from a group of items. Compare this action with a check box, which can be used to select more than one item from a group.

When you are designing the dialog box, you can set an initial value of `true` for more than one radio button, but when you open the dialog box in your application, only one can be `true` at a time.

Note: All radio buttons on the dialog box are initially grouped together and assumed to be mutually exclusive. To create multiple groups of radio buttons, see Grouping Controls.

See also:
Drawing a radio button
Radio button properties and initial values

Check box

(Standard ToolBook control.)

Use a check box when you want the user to be able to select any number of items (from none to all) from a group. Compare this action with a radio button, which can be used to select one and only one item from a group.

See also:
Check box properties and initial values
Drawing a check box

Static text

(Standard ToolBook control.)

Use the static text tool to place text, such as labels or instructions, anywhere on the dialog box. The text is displayed in a rectangle that can be any size, with or without a border. The text cannot be changed by the user.

See also:
Adding static text
Static text properties and initial values

Editable text

(Standard ToolBook control.)

Use the editable text tool to place text anywhere on the dialog box and to provide a place for users to type long text entries. The text can be changed by the user. The text is displayed in a rectangle that can be any size, with or without scroll bars. The text can be displayed in a single line or multiple lines; wordwrap can be turned on or off; typed text can be displayed as asterisks, as in password entry; and user entries can be restricted to integers only.

See also:
Adding editable text
Editable text properties and initial values

List box

(Standard ToolBook control.)

Use the list box to present a list of options to the user. The user can select one or more of the options. Users cannot type their own entries. Compare this action with a combo box.

You can allow single selection, multiple selection, or extended selection, and you can have the list sorted automatically. If the list is longer than the box, it can be scrolled.

See also:
Drawing a list box
List box properties and initial values

Combo box

(Standard ToolBook control.)

Use the combo box to present a list of options to the user. The combo box appears as a single-item box with an arrow at the right end. The user can accept the value in the box or type a value to be matched against the items in the drop-down list. The user can also click the arrow to present a list of additional choices. Compare this action with a list box.

You can have the list sorted automatically, and you can display scroll bars with the list.

See also:
Combo box properties and initial values
Drawing a combo box

Custom control

(Windows dialog box control.)

A custom control can be any Windows control, including the standard ones on the tool palette. For example, the tool bar in Word for Windows is a custom control. Custom controls are written in C or another language and compiled into the executable file or a DLL. The bitmap control on the tool palette is a custom control compiled as part of the TB30DLG.DLL.

Before a custom control can be displayed, its class must be registered with Windows.

For more information about creating and using custom controls, refer to the Windows SDK documentation.

See also:
Custom control properties and initial values
Drawing a custom control

Group box

(Windows dialog box control.)

A group box is a graphic device only; it does not have any control function of its own. You can place controls inside a group box to indicate to the user that the controls are related. Using a group box is not the same as grouping controls.

See also:
Drawing a group box
Group box properties

Bitmap

(Windows dialog box control.)

The bitmap control is used to place a bitmap on the dialog box. The bitmap does not have any control function.

Before the bitmap can be displayed in the dialog box, it must exist as a resource in the book in which the dialog box is used. You can add a bitmap control to a dialog box without assigning a bitmap to it, but nothing will show up when you open the dialog box in an application.

The Dialog editor does not display a bitmap as it will be displayed in the application. The bitmap will be centered in the control bounding box when it is displayed in the application. If the bounding box is smaller than the bitmap, the bitmap is cropped; if it is larger, a blank space appears between the edge of the bitmap and the box. Use the Preview Dialog command on the View menu to see how the bitmap will be displayed in the application.

If you need to adjust the size of the bitmap, do it in a graphics program and save the bitmap at the size you want to display it.

See also:
Bitmap properties
Placing a bitmap

Icon

(Windows dialog box control.)

The icon control is used to place an icon, such as the familiar "i" in a circle, on the dialog box. The icon does not have any control function.

Before the icon can be displayed in the dialog box, it must exist as a resource in the book in which the dialog box will be used. You can add an icon control to a dialog box without naming a specific resource, but nothing will appear when you open the dialog box in the application.

See also:
Icon properties
Placing an icon

The Dialog editor tool bar

Click a button for more information.

See also:

Creating a dialog box
Dialog box controls
Menu commands

How to draw a control

Step-by-step

Use standard ToolBook Author-level techniques to draw, size, and move controls.

To draw a control:

1. Choose the control from the tool palette.
   When you move the cursor over the Editor window, it turns into a crosshair.
2. Click in the dialog box where you want the upper-left corner of the control to be. Hold the left mouse button down while you drag the cursor down and to the right.
   As you drag the cursor, the control expands.
3. When the control is the size you want, release the mouse button.

See also:
Creating a dialog box
Previewing the dialog box

Sizing the dialog box

Step-by-step

To change the size of the dialog box, hold down the Ctrl key while you drag the sides or corners of the box.

See also:

Creating a dialog box
Previewing the dialog box

Aligning controls

Step-by-step

To align controls:

1. Select the controls you want to align.
2. From the Control menu, choose Align Control.
3. Choose one of these commands:
   Left. Aligns the left sides of all controls with the left side of the left-most control.
   Top. Aligns the tops of all controls with the top of the top control.
   Right. Aligns the right sides of all controls with the right side of the right-most control.
   Bottom. Aligns the bottoms of all controls with the bottom of the bottom control.

See also:

Creating a dialog box
Dialog box controls
Menu commands
Previewing the dialog box

Setting tabbing order for controls

Step-by-step

The tabbing order is the order in which controls receive the focus when the user presses the Tab key. The tabbing order is governed by the layer numbers of the controls; focus goes in order from the lowest number to the highest. To change the tabbing order, change the layer numbers in the controls' Properties dialog boxes.

See also:

How to set initial values for a control
How to set properties for a control
Previewing the dialog box

Setting properties for a control

Step-by-step

To set properties for a control:

1. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
2. Type the information and make the selections required in the dialog box. When you're through, click OK.

Each type of control has its own Properties dialog box. For more information about a particular control, click the control's name below.

Pushbutton
Radio button
Check box
Static text
Editable text
List box
Combo box
Group box
Icon
Bitmap
Custom control

See also:
Creating a dialog box
Dialog box controls

How to set the initial values for a control

Step-by-step

To set initial values for a control:

1. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
2. Type the information and make the selections required in the dialog box. When you're through, click OK.

Note: When your application displays the dialog box, you can use functions of TB30DLG.DLL to have it display values different from the initial values you set in the Properties dialog box.

Each type of control has its own Properties dialog box. For more information about a particular control, click the control's name below.

Pushbutton
Radio button
Check box
Static text
Editable text
List box
Combo box
Custom control

See also:
Creating a dialog box
Dialog box controls
Displaying a dialog box from within a book

Saving a dialog box definition

Step-by-step

The dialog box definition is saved as a text file with a .DIA extension. After you save the file, you can assign it to any book or books. You can also use the file to recreate the dialog box in the Dialog editor so you can change it.

To save a dialog box definition as a text file:

1. From the File menu in the Dialog editor, Choose Save As.
   The Save As dialog box appears.
2. Type a file name and select the directory for saving the file, then click OK.
   If you do not supply a file name extension, ToolBook adds a .DIA extension and saves the dialog box definition as a text file.

The .DIA text file is made up of the values for the dialog box template, followed by a single Ctrl+A character, followed by the initial values for the controls.

Note: You do not need to save the definition as a text file before you assign the dialog box to a book.

See also:

Creating a dialog box
Displaying a dialog box from within a book
Installing a dialog box in a book

Opening a dialog box in the Dialog editor

Step-by-step

You can display an existing dialog box in the Dialog editor by opening the .DIA file that contains its definition, or you can open a dialog box that's assigned to a book.

To open a dialog box .DIA file:

1. From the File menu, choose Open.
   The Open dialog box appears.
2. In the List Files Of Type box, select Dialog template (.DIA).
3. In the Directories dialog box, choose the directory that contains the file you want to open.
4. Select the file from the Files list, then click OK.

If the dialog box frame in the Dialog editor is not empty, a message appears asking if you want to clear the frame. If you choose No, the new controls are added to the existing ones.

To open a dialog box assigned to a book:

1. From the File menu, choose Open.
   The Open dialog box appears.
2. In the List Files Of Type box, select ToolBook Book (.TBK).
3. In the Directories box, choose the directory that contains the book you want.
4. Select the book file from the Files list, then click OK.
   The Load Dialog Template dialog box appears.
5. In the Dialogs In Book dialog box, choose the name of the dialog box you want to open, then click OK.

If the dialog box frame in the Dialog editor is not empty, a message appears asking if you want to clear the frame. If you choose No, the new controls are added to the existing ones.

See also:

Assigning a dialog box to a book
Creating a dialog box
Displaying a dialog box from within a book
Saving a dialog box definition

Drawing pushbuttons

Step-by-step

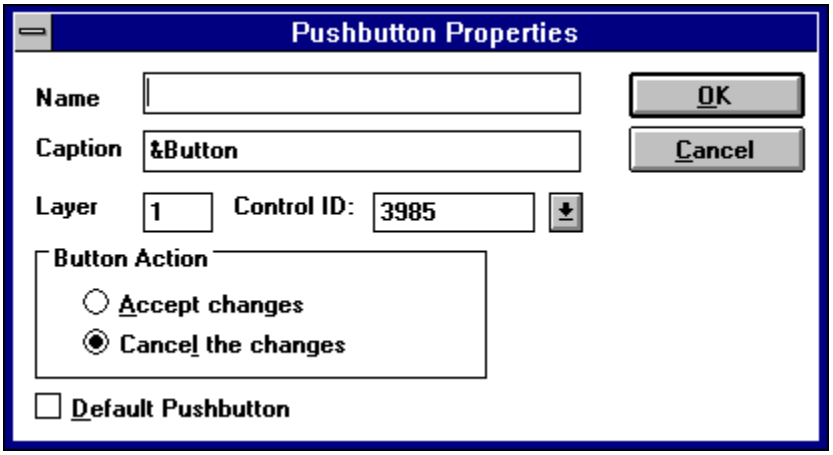
To draw a pushbutton:

1. From the tool palette, select the pushbutton control.
2. Draw the pushbutton, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. Repeat these steps to place additional pushbuttons on the dialog box.

See also:
Creating a dialog box

Pushbutton properties and initial values

Click a field or button to see more information.



See also:

Creating a dialog box

Drawing radio buttons

Step-by-step

To draw a radio button:

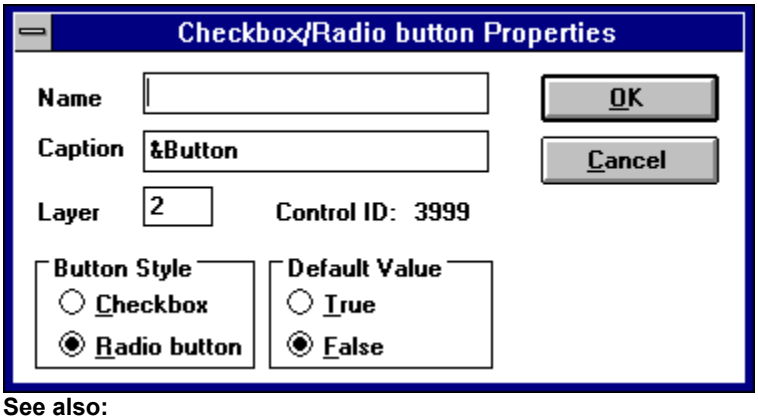
1. From the tool palette, select the radio button control.
2. Draw the radio button, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. Repeat these steps to place additional radio buttons on the dialog box.

By default, all of the radio buttons on a dialog box are mutually exclusive: only one can be selected at a time. If you want to be able to select more than one at a time, you can put the radio buttons into groups.

See also:
Creating a dialog box

Radio button properties and initial values

Click a field or button to see more information.



See also:

Changing a radio button to a check box
Creating a dialog box

Creating multiple groups of radio buttons

Step-by-step

By default, all radio buttons on a dialog box, no matter where they are located, are considered part of a single group. Thus, only one can be selected at a time.

You can use the Group command on the Control menu to group radio buttons. Then, one button in each group can be selected.

For more information about grouping controls, see Grouping Controls or refer to the ToolBook User Manual.

Note: Putting a group box around several radio buttons does not group them functionally; it only provides a visual cue that the buttons are related.

See also:
Creating a dialog box

Grouping controls

Step-by-step

By default, all of the radio buttons on a dialog box are mutually exclusive; only one can be selected at a time. If you want to be able to select more than one at a time, you can put the radio buttons in groups. (If you group one set of radio buttons, all other buttons on the dialog box automatically function as it they were in another group.)

To place radio buttons in a group:

1. Select all of the radio buttons you want to include in the group.
2. From the Control menu, choose Group.

   The bounding boxes around the individual buttons are replaced with a single bounding box that encloses all of the buttons in the group.

Note: The group box control does not perform the grouping function. It simply draws a border around selected controls as a visual indication that they are related.

See also:

Creating a dialog box

Changing a check box to a radio button
Changing a radio button to a check box

Step-by-step

To change a check box to a radio button, or vice versa, change the control's button style property. For more information, see Radio button properties or Check box properties.

See also:

Creating a dialog box
Drawing check boxes
Drawing radio buttons

Drawing check boxes

Step-by-step

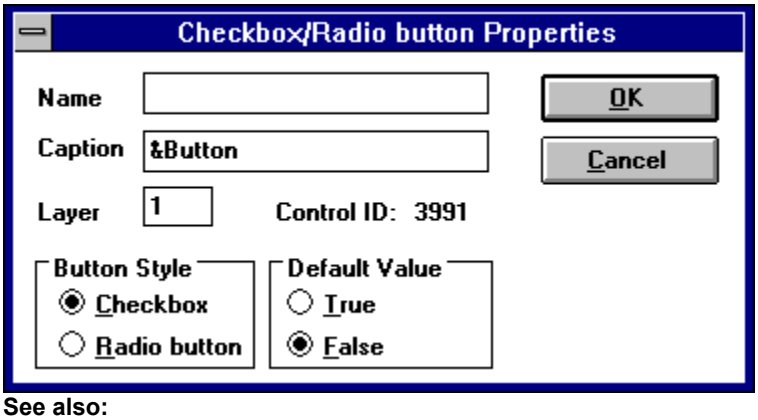
To draw a check box:

1. From the tool palette, select the check box control.
2. Draw the checkbox, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. Repeat these steps to place additional check boxes on the dialog box.

See also:
Creating a dialog box

Check box properties and initial values

Click a field or button to see more information.



See also:

Changing a check box to a radio button
Creating a dialog box

Adding static text

Step-by-step

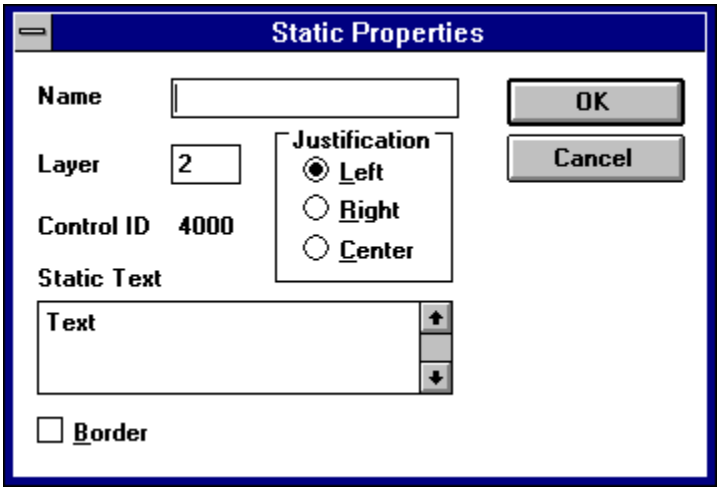
To add static text:

1. From the tool palette, select the static text control.
2. Draw the static text control, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. After you type the text for the box, you may need to adjust its size and shape again.
6. Repeat these steps to place additional static text controls on the dialog box.

See also:
Creating a dialog box

Static text properties and initial values

Click a field or button to see more information.



See also:

Creating a dialog box

Adding editable text

Step-by-step

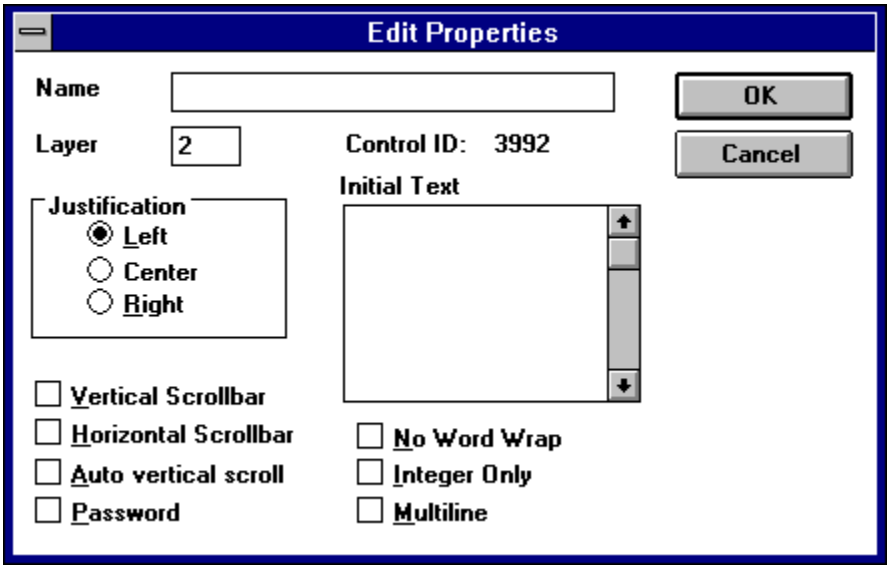
To add editable text:

1. From the tool palette, select the editable text control.
2. Draw the editable text control, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. After you type the text for the box, you may need to adjust its size and shape again.
6. Repeat these steps to place additional editable text controls on the dialog box.

See also:
Creating a dialog box

Editable text properties and initial values

Click a field or button to see more information.



See also:

Creating a dialog box

Drawing a list box

Step-by-step

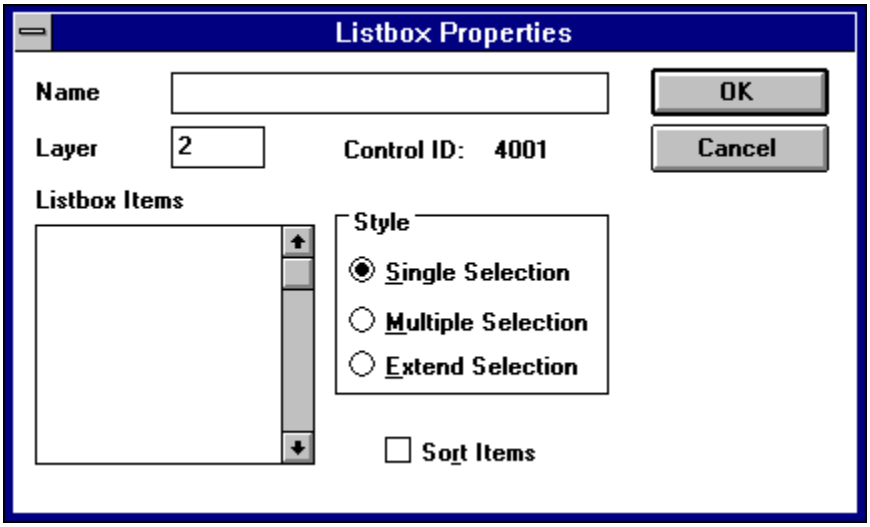
To draw a list box:

1. From the tool palette, select the list box control.
2. Draw the list box control, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. After you type the text for the box, you may need to adjust its size and shape again.
6. Repeat these steps to place additional list box controls on the dialog box.

See also:
Creating a dialog box

List box properties and initial values

Click a field or button to see more information.



See also:

Creating a dialog box

Drawing a combo box

Step-by-step

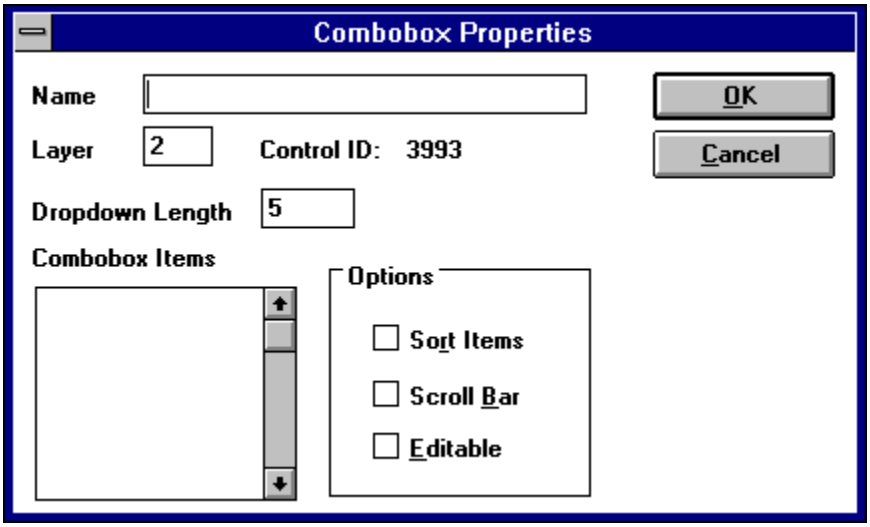
To draw a combo box:

1. From the tool palette, select the combo box control.
2. Draw the combo box control, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. After you type the text for the combo box, you may need to adjust its size and shape again.
6. Repeat these steps to place additional combo box controls on the dialog box.

See also:
Creating a dialog box

Combo box properties and initial values

Click a field or button to see more information.



See also:

Creating a dialog box

Drawing a custom control

Step-by-step

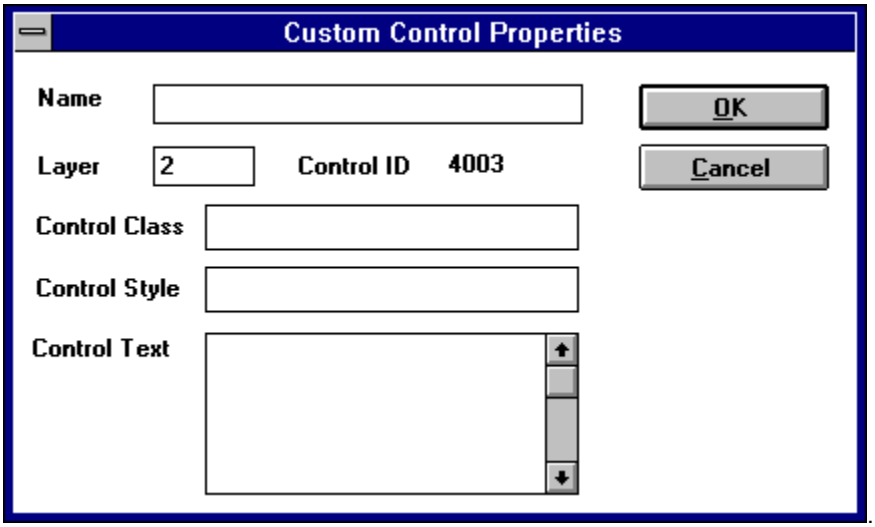
To draw a custom control:

1. From the tool palette, select the custom control.
2. Draw the custom control, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties and initial values for the control. When you're through, click OK.
5. Repeat these steps to place additional custom controls on the dialog box.

See also:
Creating a dialog box

Custom control properties and initial values

Click a field or button to see more information.



See also:

Creating a dialog box

Drawing a group box

Step-by-step

To draw a group box:

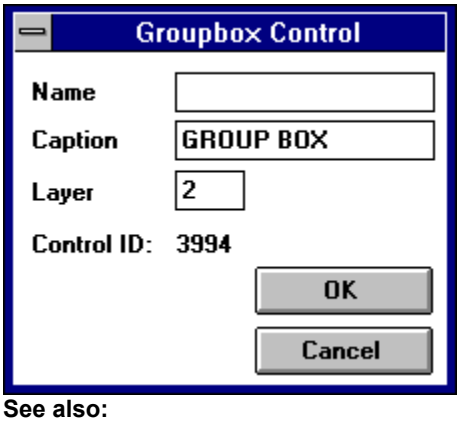
1. From the tool palette, select the group box control.
2. Draw the group box control, using standard ToolBook drawing procedures.
   The group box grows as you move the cursor. The group box is opaque, so it hides any controls that are behind it.
3. To reveal the controls behind the group box, use the Send To Back button on the tool bar.
4. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
5. Set the properties for the control. When you're through, click OK.
6. Repeat these steps to place additional group box controls on the dialog box.

Note: Putting a group box around radio buttons does not group them functionally; it only provides a visual cue that the buttons are related. For more information about grouping controls, see Grouping controls.

See also:
Creating a dialog box

Group box properties

Click a field or button to see more information.



See also:

Creating a dialog box

Placing a bitmap

Step-by-step

To place a bitmap:

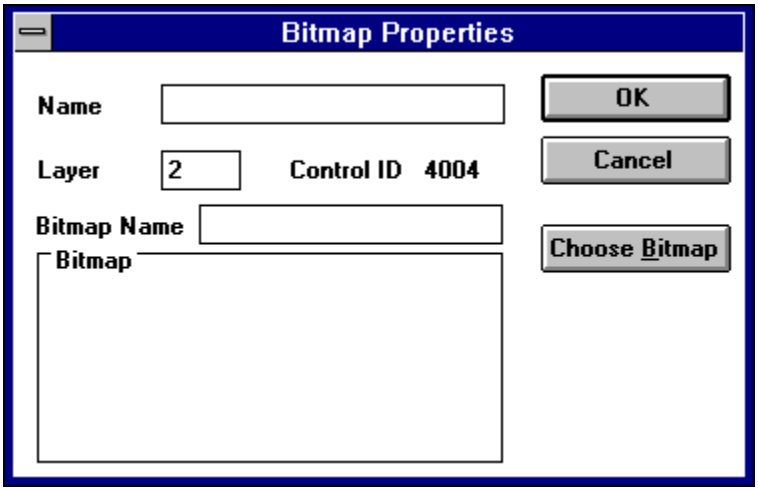
1. From the tool palette, select the bitmap control.
2. Draw the bitmap control, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties for the control. When you're through, click OK.
5. Repeat these steps to place additional bitmap controls on the dialog box.

See also:

Creating a dialog box

Bitmap properties

Click a field or button to see more information.



See also:

Creating a dialog box

Placing an icon

Step-by-step

To place an icon:

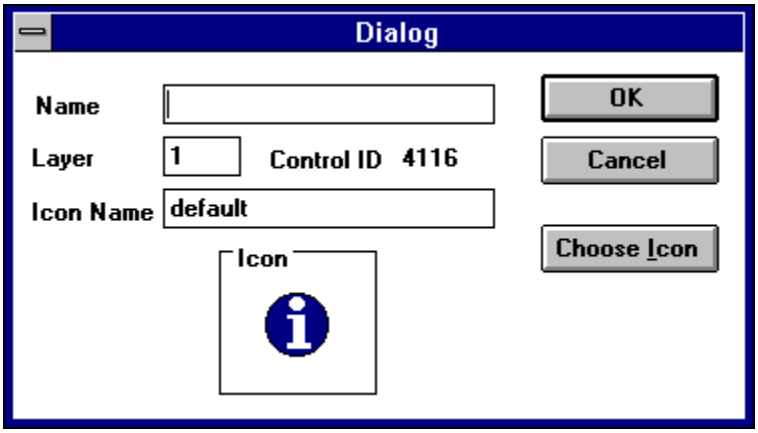
1. From the tool palette, select the icon control.
2. Draw the icon control, using standard ToolBook drawing procedures.
3. Double-click the control with the left mouse button.
   The Properties dialog box for the control appears.
4. Set the properties for the control. When you're through, click OK.
5. Repeat these steps to place additional icon controls on the dialog box.

See also:

Creating a dialog box

Icon properties

Click a field or button to see more information.

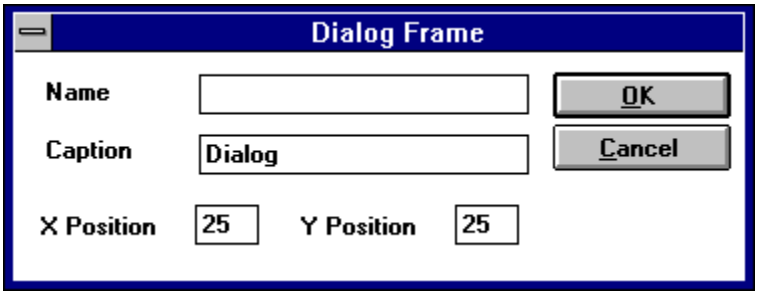


See also:

Creating a dialog box

Dialog frame properties

Click a field or button to see more information.



See also:

Creating a dialog box

OK button

Click this button to accept the changes and close the Properties dialog box.

Icon name box

The name of the icon resource in the book in which the dialog box is used.

Default value

Radio buttons and check boxes can have default values of `true` or `false`. For `true`, the button is filled or checked when the dialog box is displayed; for false, the button is empty.

Note that you can set `true` as the default value for more than one radio button, but only one is shown `true` when the application opens the dialog box.

Combo box items

The items that appear on the drop-down menu. If you precede one of the items with an ampersand (&), it appears in the box when the dialog box opens. Otherwise, the box is empty.

Layer number

Layer numbers control the tabbing order among controls (the control with the lowest number is first) and which control is visible when two overlap (the one with the higher number obscures the other). Layer numbers are automatically assigned to correspond with the order in which controls are placed on the dialog box. You can change them to control tabbing order and appearance.

Bring To Front button

Click this button to move the selected control to the front. Equivalent to setting the control's layer number to the highest of all objects on the dialog box. If the control overlaps other controls, it obscures them.

Multiline

Check this box to allow the text to wrap within the width of the text box. If this box is not checked, the text appears on only one line, regardless of how deep the box is, and disappears off the right side.

Auto vertical scroll

Check to have the text scroll vertically when the user drags the cursor past the upper or lower edge of the text box. This feature does not work if Multiline is not selected.

Send To Back

Moves the selected control to the back. Equivalent to setting its layer number to the lowest of all objects on the dialog box. If the control overlaps other controls, they obscure it.

Bitmap

The bitmap to be placed in the dialog box.

Icon

The icon to be placed on the dialog box.

Save file

Saves the current dialog box template as a text file. If the template has not been saved, a dialog box appears, in which you can give the file a name.

Button style

Choose check box or radio button. Note that radio buttons in the same group are exclusive (only one can be `true` at a time), while check boxes are not.

Open file

Opens the Open dialog box, in which you can select a directory and a file to open.

Justification

Choose how text will be aligned in the box:

- Left. Even on the left and ragged on the right
- Right. Even on the right and ragged on the left
- Center. Symmetric about the center of the box, ragged on the left and right

No wordwrap

Check this box to prevent words from wrapping to the next line automatically. To go to the next line, you must enter a carriage return.

Duplicate

Places a copy of the selected control on the dialog box.

Password

When you check this box, only s appear when the user types an entry in the box. This feature is usually used when the user must enter a password, so that the password itself does not appear onscreen.

Undo/Redo

Choose to undo the most recent action. Choose again to redo that action.

Dropdown length

The number of lines that appear when the list is dropped. Only whole lines are displayed.

Bitmap name

The name of the bitmap resource in the book in which the dialog box is used. Once you choose a resource, you must give it a name, or use the name it already has.

Options

Check the boxes for the options you want:

- Sort items. Sorts the items in the list.
- Scroll bar. Displays a vertical scroll bar on the right side of the list.
- Editable. Allows the user to type a new value or edit the value in the box. As the user types in the box, the list scrolls to the nearest match to what is being typed.

Choose icon

Click this button to open the Choose Resource dialog box, in which you can choose the icon you want to use in the dialog box.

Control class

The Windows class for the control.

Character format

Opens the Character dialog box, in which you can select the type face, size, and other attributes.

Sort items

Check this box to have the items in the list box list sorted in ascending order.

Border

Check to add a border to the text box.

Name

A name you can assign to the control for use in scripts and handlers. Optional.

Style

The type of selection the user can make:

- Single. Only one item can be selected.
- Multiple. User can select multiple items, which do not need to be contiguous.
- Extend. User can select one item, then drag the cursor up or down to select other contiguous items. Or, the user can click one item, then click another item to select all items between and including the two.

Open properties box

Opens the Properties dialog box for the selected control. Equivalent to double-clicking the control.

Static text

Type or paste the text you want to appear on the dialog box. You can enter up to 32,000 characters.

Tip: Don't press the Enter or Return key when you come to the right side of the box, or the text will not be able to adjust itself if you change the dimensions of the box.

List box items

The items from which the user can choose.

Vertical scrollbar

Check this box to have a vertical scroll bar added to the right side of the text box. The scroll bar is added regardless of whether the text is longer than the box. The scroll bar does not appear when you are designing the dialog box, but does appear in preview. It is located inside the border of the text box.

Control style

The Windows control style. Must be a decimal value. For more information, refer to the Windows SDK documentation.

Control text

Any text that is required for the control, such as a label on a button.

Cancel

Choose this button to cancel the changes and close the Properties dialog box.

Integer only

Check this box to prevent the user from entering anything but integers. You can still enter alphabetic characters in the initial text, but the user cannot.

Preview

Shows you how the dialog box will look in the application. You can choose 3-D or normal appearance.

Initial text

Type or paste the text you want to appear when the dialog box is displayed. Optional. The user can edit this text. You can type up to 32,000 characters.

Tip: Don't press the Enter or Return key when you come to the right side of the box, or the text will not be able to adjust itself if you change the dimensions of the box.

Horizontal scrollbar

Check this box to have a horizontal scroll bar added to the bottom of the text box. The scroll bar is added regardless of whether the text is wider than the box. The scroll bar does not appear when you are designing the dialog box, but does appear in preview. It is located inside the border of the text box.

If you select No Word Wrap, you'll need a horizontal scroll bar, or text will disappear beyond the right side of the text box.

Button action

The action that takes place when the button is chosen. Changes in the dialog box are accepted and the box is closed, or changes are canceled and the box is closed.

These are the inherent functions of the pushbutton. If you want the button to do something else, use the callback function. See Displaying a dialog box from within a book.

Default pushbutton

When this box is checked, this button is activated when the user presses Enter. Only one button can have this box checked.

Control ID number

The ID number assigned automatically. You can refer to the control's ID number in scripts and handlers.

For pushbuttons only: When ID_OK is selected in the Control ID combo box, the button is activated when the user presses the Enter key; when ID_CANCEL is selected, the button is activated when the user presses the Esc key.

Caption

A label that appears on the button. Optional.

If you type an ampersand (&) before a letter in the caption, that letter is underlined on the button, and the button can be activated by holding down the Alt key and typing the letter.

Caption

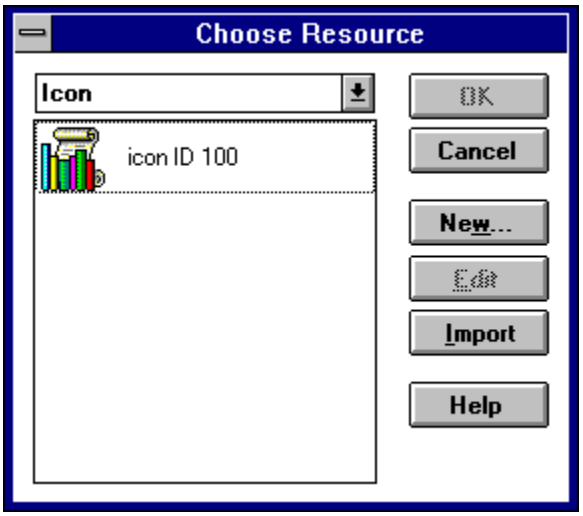
A label that appears in the top border of the box. Optional.

Choose bitmap

Click this button to open the Choose Resource dialog box, in which you can choose the bitmap you want to use in the dialog box.

Choose Resource

Click a field or button to see more information.



Position

The position of the dialog box relative to upper left corner of its parent. If it can't be in this position because of the location of the parent, it fits on the screen close as possible.

Size

The size of the control's bounding box, measured in Windows dialog units.

X position

The horizontal position of the upper-left corner of the dialog box relative to the upper-left corner of the window. The position is measured in Windows dialog units.

Y position

The vertical position of the upper-left corner of the dialog box relative to the upper-left corner of the window. The position is measured in Windows dialog units.

Handler example

The `show<name>` handler links the functions in the TB30DLG.DLL, sets the initial values for the dialog box, and displays the dialog box.

```
to handle showDepartim
  --Move linkDLL statement to enterBook handler
  linkDLL "TB30DLG.dll"
    string dialog(string, string)
    string setValue(string, string, string)
    string getValue(string, string)
  end

  set init to DepartimInit of this book
  --set init to setValue(init,"","") --DLL function

  set retValue to dialog(DepartimBOX of this book, init)
  --get getValue(retValue, "") --DLL function
end
```

For more information, see Displaying a dialog box from within a book.

Previewing the dialog box

From the View menu, choose Preview Dialog. You can select 3-D or normal appearance.

Assigning a caption

Step-by-step

To assign a caption to a dialog box:

1. Click the dialog box title bar to select the dialog box frame.
2. Open the Properties dialog box for the dialog box frame.
3. Type the caption in the Caption box.
4. Click OK.

Positioning the dialog box

Step-by-step

To position the dialog box:

1. Click the dialog box title bar to select the dialog box frame.
2. Open the Properties dialog box for the dialog box frame.
3. Type the X position and Y position in their respective boxes .
4. Click OK.

Windows dialog unit

A horizontal Windows dialog unit is one-fourth the width of a character of the type face and size used for the dialog box. A vertical Windows dialog unit is one-eighth of the height of a character.

Note: The equivalent measurement in inches or centimeters depends on the display device being used.

New

Opens a new, blank dialog box frame. If you already have a dialog box open, it will be closed. Make sure you save it before you open a new dialog box.

Open

Opens the Open dialog box, in which you can select a file to open.

Save

Saves the dialog box as a .DIA file. This command is disabled if the dialog box is empty.

If the dialog box has not already been saved, the Save dialog box opens so you can give the file a name.

Save As

Opens the Save As dialog box, so you can give the file a name and save it as a .DIA file.
This command is disabled if the dialog box is empty.

Assign Template To Book

Opens the Choose Target Book For Dialog Template dialog box, in which you can select the book to which you want to assign the template. After you select the book and click OK, you are asked for a name for the dialog box. A message then appears to confirm that the template has been assigned to the book you selected. The message also gives you the name of the handler and tells you how to display the dialog box in the book.

Convert template to viewer

Allows you to convert a dialog box created in the Dialog editor into a ToolBook page and a viewer to display it.

Exit

Closes the Dialog editor. If you have not saved the changes to the current dialog box, you are prompted to do so.

Undo/Redo

Choose this command before you do anything else to reverse the effect of your most recent action. After you undo an action, the command changes to Redo. Chose Redo before you do anything else to reinstate your last action.

Some actions (such as drawing a control) cannot be undone; in that case, the command changes to Cannot Undo.

Cut

Removes the selected object from the dialog box and places it on the Clipboard. The object can be pasted into the same or another dialog box.

Copy

Places a copy of the selected object on the Clipboard, leaving the original in place. The copy can be pasted into the same or another dialog box.

Paste

Places the contents of the Clipboard into the open dialog box. If you select a control, choose Copy, and then choose Paste, the new control is pasted directly over the original one.

Delete

Removes the selected control from the dialog box. The control is not placed on the Clipboard and cannot be retrieved; however, you can undo the action.

Clear Dialog

Removes all controls from the dialog box and restores the dialog box to its default size.

Properties

Opens the Properties dialog box for the selected control.

Group

Places the selected radio buttons in a group. Controls in the group are mutually exclusive; only one can be selected at a time.

If you select a control that is already in a group, the command changes to Ungroup. Choosing Ungroup removes all controls from the group.

Align Controls

Aligns the selected controls with each other. For more information, see Aligning Controls.

Preview Dialog

Shows how the dialog box and controls will look and work in the application.

3D in Preview

Applies the Windows 3D style to the dialog box when you preview it.

Quick Help

Opens a help window that gives brief instructions for some of the most common Dialog editor tasks.

Edit Editor

Untranslates all Windows messages, enabling right-click functionality and Command Window access. We recommend that you not change any of the scripts.

About Dialog

Displays the About Dialog screen.

Initialization values

A text string that contains the initial values that are placed in the controls on the dialog box when it is displayed. The Dialog editor places this string in a book user property when you assign the dialog box to a book.

Dialog box template

A text string that contains the specifications Windows uses to draw the dialog box. The Dialog editor places this string in a book user property when you assign the dialog box to a book.

Properties box

A dialog box in which you can set various properties for a control (and for the dialog box frame). Each kind of control has different properties. For more information about controls and their properties, see dialog box controls.

dlgBox

The book user property that contains the dialog box template. This data is passed to Windows when the dialog box is displayed, and Windows uses it to draw the dialog box.

dlgInit

The book user property that contains the initial values for the controls on a dialog box. This data is passed to Windows when the dialog box is displayed, unless it is replaced by other initial values at that time.

Window handle

The window handle of the window calling the dialog box.

Notify object

The target object for the `to get TBKDialogInit`, `to get TBKDialogCommand`, and `to get TBKDialogDestroy` handlers.

Edit button

Choose an icon or bitmap from the resource list box, then choose this button to edit it. An editor for the icon or bitmap appears.

Resource list

A list of all icons or bitmaps available in the book to which you are assigning the dialog box.

Resource combo box

The only choice available is the type whose control you are adding to the dialog box; that is, either icon or bitmap.

Import button

Choose this button to import a bitmap or icon from another book.

Help button

Choose this button for an explanation of the dialog box controls.

New button

Choose this button if you want to create a new icon or bitmap. An editor appears in which you can create the object.

Installing a dialog box in a book

Step-by-step

To install a dialog box in an application:

1. Start the Dialog editor and open the dialog box.
2. On the File menu, choose Assign Template to Book.
   The Choose target book for dialog template dialog box appears.
3. Select the book in which you want to install the dialog box, then click OK.
   You are prompted to give the dialog box a name.
4. Type the name in the box, then click OK.
   A message appears telling you that the dialog box template and initialization values have been added to the book as a user property and that a handler for the message show<dialog box name> has been added to the book script.

See also:

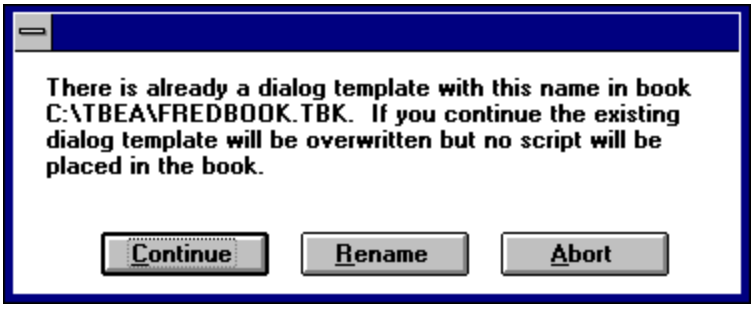
Creating a viewer from the dialog definition
Displaying a dialog box from within a book
Revising dialog box definitions

Revising dialog box definitions

Step-by-step

To revise a dialog box:

1. Open the dialog box in the Dialog editor.
2. Make your changes.
3. Save the .DIA file.
4. From the File menu, choose Assign template to book. Assign the dialog box to the book again.
   If the dialog box has already been assigned to the book, the following dialog box appears:



5. To assign the dialog box with a different name (and retain the original dialog box in the book), click the Rename button. Type a new name, then click OK.
   To assign the dialog box with the same name as before, click Continue.
   A message appears telling you the dialog box has been assigned to the book.
6. Click OK.

See also:

Displaying a dialog box from within a book
Installing a dialog box in a book
Opening a dialog box definition

Creating a viewer from the dialog box definition

Step-by-step

To create a viewer from the dialog box definition:

1. Create the dialog box, or open the dialog box .DIA file in the Dialog editor.
2. From the File menu, choose Convert template to viewer.
   The Choose Target Book For Dialog Viewer dialog box appears.
3. Select the book in which you want to create the viewer, then click OK.
   A message appears informing you that dialog viewer <name> and handler show<name> have been created in the book.

See also:

Displaying a dialog box from within a book

Displaying a dialog box from within a book

Step-by-step

When you assign the dialog box to a book, a handler (example) is added to the book script to handle the message `show<name>`, where name is the name of the dialog box. When the handler is called, it

- links to the tb30dlg.dll functions.
- displays the dialog box with the initial values you set when you created the box.
- gets the values in the dialog box when the user closes it.

This handler displays a static dialog box, which is out of the application's control while it is open. You can also use a dialog box interactively, in which case the application can find out what the user does and make changes to the dialog box while it is open. To use the dialog box interactively, you'll need additional handlers to respond to the actions the user takes. For details, see Static versus interactive dialog box.

To display the dialog box, send the message `show<name>` to the book.

The usual way to send the `show<name>` message is to add a button to your book with a script like the following (in this example, to display a dialog box named "Routes"):

```
to handle buttonDown
  send showRoutes
end
```

You don't have to use a button to display the dialog box; any ToolBook event that sends a message can be used. For example, to have a password dialog box appear as soon as the book opens (when the `enterApplication` message is sent), you could add a script like this to the book:

```
to handle enterApplication
  send showPass
end
```

This displays a dialog box named Pass, in which the user types a password.

You can also display a dialog box with a menu command.

Note: Make sure `sysLockScreen` is set to false before displaying a dialog box created with the Dialog editor. Otherwise, the ToolBook window will not redraw properly when the user moves the dialog box or makes the window inactive.

See also:

Displaying a static dialog box -- dialog( )
Displaying an interactive dialog box -- dialogCallback( )

Displaying a dialog box from a menu

Step-by-step

Use the `add menuItem` command to add a command to a menu to display the dialog box. Give the menu command the alias `show<dialog box name>`.

When the user chooses the command from the menu, ToolBook sends the `show<dialog box name>` and the dialog box appears.

For details about creating menus, refer to Chapter 13 of the ToolBook User Manual.

Note: Make sure `sysLockScreen` is set to `false` before displaying a dialog box created with the Dialog editor. Otherwise, the ToolBook window will not redraw properly when the user moves the dialog box or makes the window inactive.

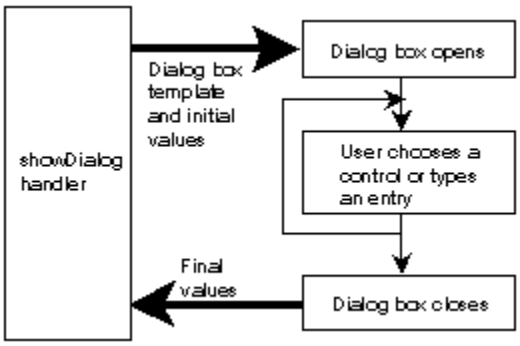
See also:
Displaying a dialog box from within a book

Static versus interactive dialog box

Step-by-step

A dialog box can be used in either a static or an interactive mode. For either mode, the design is the same, so the same dialog box can be used both ways. To display the dialog in a static mode, you call the `dialog( )` function of TB30DLG.DLL. To display an interactive dialog box, call `dialogCallback( )`.

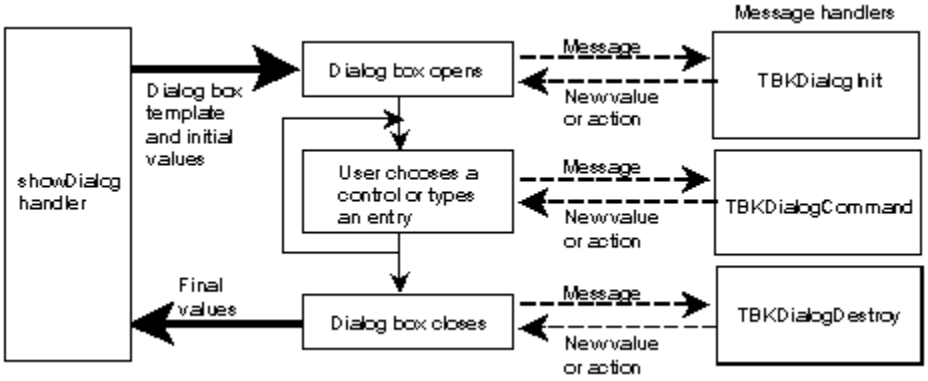
Used in a static mode, the dialog box accepts the user's choices or typed entries in its controls. Other than that, it does not change in response to what the user does. When the user closes the dialog box, the choices and entries they made are passed back to the handler that opened the dialog box.



In interactive mode, the dialog box sends messages back to the object that opened it

- when it opens.
- when the user makes a choice or types an entry in it.
- when it closes.

By using handlers to respond to these messages, you can change the dialog box in response to what the user does.



For example, your dialog box could contain a group of radio buttons labeled "plane," "bus," and "train" and a list box labeled "departure times." When the user chooses a method of transportation, the entries in the list box could change to those for the chosen method. Or, a user could be asked to enter a password, and then the program could check to see if the entry was correct before enabling the remaining controls in the dialog box.

In either mode, you can change the initial values of the dialog box control before you open the dialog box, and you can get the values that result from the choices the user makes and the entries they type. In both modes, you use handlers to display and control the dialog box.

See also:

Dialog box message handlers
Displaying a dialog box from within a book
Using dialogCallback( ) functions
Windows notification messages

Handlers

Step-by-step

To use a dialog box in either a static or an interactive mode, you need handlers. At the minimum, you will need a handler to link the dialog DLL, TB30DLG.DLL, and declare the DLL functions you want to use, and another handler that displays the dialog box.

The Dialog editor creates a basic handler to link the DLL and a basic handler that calls `dialog()` to display the static dialog box. That script and handler are placed in the book to which you assign the dialog box.

Here's an example of a script and handler (for the dialog box Routes) created by the Dialog editor application:

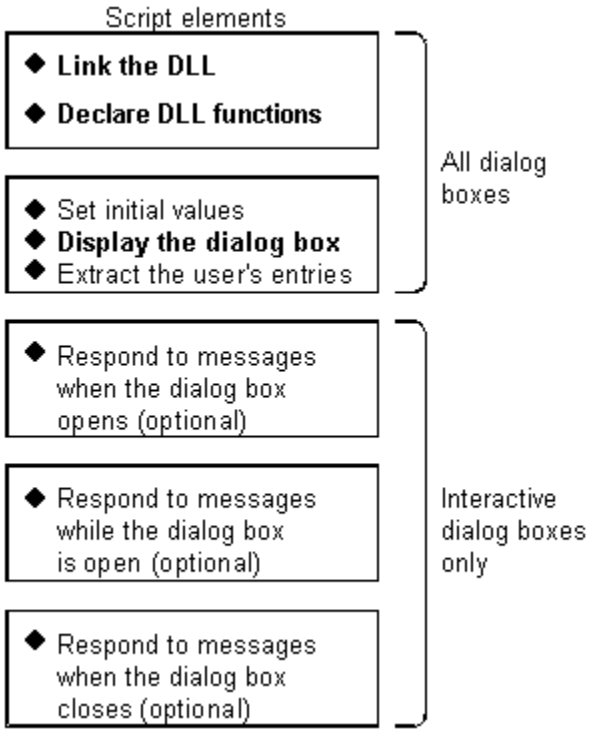
```
to handle showRoutes
  --Move linkDLL statement to enterBook handler
  linkDLL "tb30dlg.dll"
    string dialog(string, string)
    string setValue(string, string, string)
    string getValue(string, string)
  end

  set init to RoutesInit of this book
  --set init to setValue(init,"","")  --DLL function

  set retValue to dialog(RoutesBOX of this book, init)
  --get getValue(retValue, "")  --DLL function
end
```

For a static dialog box, you should add instructions for using the values in the controls after the user closes the dialog box. Those instructions belong in the handler that displays the dialog box, after `get getValue(retValue, "")`.

To use a dialog box in interactive mode, you will need to modify the linking script and the handler, and you will need to add at least one script to respond to messages from the dialog box. For further information about interactive-mode scripts, see Displaying an interactive dialog box.



See also:

Displaying a static dialog box -- dialog( ) example

Step-by-step

Use the dialog( ) function in TB30DLG.DLL to display a static dialog box.

```
to handle showDepartim
  --Move linkDLL statement to enterBook handler
  linkDLL "TB30DLG.dll"
    string dialog(string, string)
    string setValue(string, string, string)
    string getValue(string, string)
  end

  set init to DepartimInit of this book
  --set init to setValue(init,"","")  --DLL function

  set retValue to dialog(DepartimBox of this book, init)
  --get getValue(retValue, "")  --DLL function
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Displaying an interactive dialog box -- dialogCallback( ) example

Step-by-step

The following is an example of the script elements for an interactive dialog box. The first handler links and initializes several functions in TB30DLG.DLL. Then it calls `dialogCallback( )` to display the dialog box. Additional handlers respond to messages from the dialog box. In this example, when the user chooses a mode of transportation, the labels for the Depart From buttons change and the departure times in the list box change.

```
to handle showDepartim
  --Move linkDLL statement to enterBook handler
  linkDLL "TB30DLG.dll"
    string dialogCallback(word, string, string, string)
    string setValue(string, string, string)
    string getValue(string, string)
    string getControlText (word, string)
    int setControlText (word, string, string)
    int setListBoxItems (word, string, string)
  end

  set init to DepartimInit of this book
  --set init to setValue(init,"","")  --DLL function

  set retValue to dialogCallback( windowHandle of this window, \
  DepartimBox of this book, init, self)
  --get getValue(retValue, "")  --DLL function
end


-- handlers for notification messages sent by dialog call back

to get tbkDialogInit hDlg
  --when dialog is created (WM_INITDIALOG)
  return 1
end

to get tbkDialogCommand  hDlg, CtrlID, hWndCtrl, Msg, CtrlName
  conditions
  when CtrlName = "BUTTON air_button"
    get setControlText( hDlg, "BUTTON city_one", "San Francisco" )
    get setControlText( hDlg, "BUTTON city_two", "Oakland" )
    get setControlText( hDlg, "BUTTON city_three", "San Jose" )
    get setListBoxItems( hDlg, "LISTBOX depart_times", \
    "7 a.m."&crlf&"8:30 a.m."&crlf&"10 a.m."&crlf& \
    "12 noon"&crlf&"1:30 p.m."&crlf&"3 p.m."&crlf& \
    "5 p.m."&crlf&"7 p.m.")

  when CtrlName = "BUTTON Train_button"
    get setControlText( hDlg, "BUTTON city_one", "Hayward" )
    get setControlText( hDlg, "BUTTON city_two", "Santa Cruz" )
    get setControlText( hDlg, "BUTTON city_three", "Santa Rosa" )
    get setListBoxItems( hDlg, "LISTBOX depart_times", \
    "7:30 a.m."&crlf&10 a.m."&crlf&"5 p.m.")
  end conditions
  return 1
end

to get tbkdialogDestroy
```

```
    --(WM_DESTROY)
    return 1
  end
```

See also:
Dialog box message handlers
Displaying a dialog box from within a book
Using dialogCallback( ) functions
Windows notification messages

Setting initial values

Step-by-step

The values you entered when you created the controls in the dialog box are contained in the dlgInit string. When you assign the dialog box to a book, this string is stored as a user property of the book. The `show<name>` handler passes this string to Windows when the dialog box opens.

If you want to open the dialog box with different initial values, you can change them with the `setValue` function.

```
to handle showDepartim
  --Move linkDLL statement to enterBook handler
  linkDLL "TB30DLG.dll"
    string dialog(string, string)
    string setValue(string, string, string)
    string getValue(string, string)
  end

  set init to DepartimInit of this book
  --set init to setValue(init,"city_one","San Francisco")  --DLL function

  set retValue to dialog(DepartimBOX of this book, init)
  --get getValue(retValue, "")  --DLL function
end
```

Note: If you're using the dialog box interactively, you can use a `to get tbkDialogInit` handler instead of the `setValue( )` function to change initial values before the dialog box is displayed.

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Determining what the user entered in a static dialog box

Step-by-step

With a static dialog box, you find out what the user entered after the dialog box closes. Then the return value (`retValue`) contains the values of all controls at the time the dialog box closed.

You can use the `getValue` function to find the values of particular controls.

```
to handle showDepartim
  --Move linkDLL statement to enterBook handler
  linkDLL "TB30DLG.dll"
    string dialog(string, string)
    string setValue(string, string, string)
    string getValue(string, string)
  end

  set init to DepartimInit of this book
  --set init to setValue(init,"","")  --DLL function

  set retValue to dialog(DepartimBOX of this book, init)

  get getValue(retValue, "nListBox depart_times")  --DLL function
  request "Selected item number" && It
end
```

See also:

Determining what the user enters in an interactive dialog box
Displaying a dialog box from within a book
Static versus interactive dialog box

Determining what the user enters in an interactive dialog box

Step-by-step

With an interactive dialog box, you can find out what the user chooses or enters immediately, while the dialog box is still open.

You do that by writing callback message handlers that respond to messages from the dialog box. The handlers can respond to specific actions such as the user's clicking on a particular button or scroll bar.

See also:

Determining what the user entered in a static dialog box
Displaying a dialog box from within a book
Static versus interactive dialog box
Using dialogCallback( ) functions
Windows notification messages

Callback message handlers

Step-by-step

When a dialog box is used interactively, it sends messages at three different times:

- When the dialog box opens (WM_INITDIALOG)
- When the user takes some action in the dialog box (WM_COMMAND)
- When the dialog box closes (WM_DESTROY)

To receive these messages, you use three "to get" handlers:

- `to get TBKDialogInit` (for messages when the dialog box opens)
- `to get TBKDialogCommand` (for messages when the user takes some action in the dialog box)
- `to get TBKDialogDestroy` (for messages when the dialog box closes)

You only need to write handlers for the kinds of messages on which you want to act.

A user's action often results in more than one notification message being sent. For example, when the user clicks in the edit field of a combo box, three messages are sent:

- 10 -- The user's selection should be canceled.
- 4 -- The list box is receiving the input focus.
- 3 -- The combo box is receiving the input focus.

Your message handler can respond to a specific action identified by a notification message, or it can simply respond to the fact that an action has taken place, regardless of what it is.

See also:

Determining what the user enters in an interactive dialog box
Displaying a dialog box from within a book
Responding to a specific action
Responding to Escape and Enter keys
Using dialogCallback( ) functions
Windows notification messages

Responding to a specific action (dialogCallback( ) only)

Step-by-step

To respond to a specific action, the handler should check the message number that is passed with the notification message and take action when the correct message is received:

```
to get tbkDialogCommand hDlg, CtrlID, hWndCtrl, Msg, CtrlName
if msg = 0 and CtrlName = "Button Help"  -- msg 0 = BN_CLICKED
  show viewer "QuickHelp" as modal
  if isOpen of viewer "QuickHelp"
    close viewer "QuickHelp"
  end
  return 1                                    -- return 1 so dialog box
                                              -- doesn't close
else
  return 0
end
```

See also:

Controlling default behavior of controls
Determining what the user enters in an interactive dialog box
Displaying a dialog box from within a book
Responding to Escape and Enter keys
Using dialogCallback( ) functions
Windows notification messages

Controlling default behavior of dialog box windows

Step-by-step

These actions cause the dialog box window to close:

- Clicking any pushbutton on the dialog box
- Double-clicking a single-select list box
- Pressing the Escape key on the keyboard
- Pressing the Enter key on the keyboard

When you are using a dialog box interactively, you can override these default actions.

Note: You cannot override the default action of a control. For example, the default action of a pushbutton is to appear pressed when the user clicks it and to return to its normal appearance when the user releases the mouse button. You cannot override that action, but you can prevent the dialog box from closing after the button is clicked.

To prevent the dialog box from closing when one of these events takes place, write a callback message handler that responds to the event. The handler can perform whatever action you want, or it can perform no action. Return 0 from the handler. (Returning a value of 1 allows the default behavior to occur.)

See also:

Determining what the user enters in an interactive dialog box
Displaying a dialog box from within a book
Respond to a specific action
Responding to Escape and Enter keys
Using dialogCallback( ) functions
Windows notification messages

Using dialogCallback( ) functions

Step-by-step

The following are operations you can perform using the functions of the TB30DLG.DLL. Click an operation to see a sample script.

Note: Before you can call a DLL function, you must declare it in the handler that links the DLL.

Adding or deleting items in a combo box quickly

Adding or deleting items in a list box quickly

Changing an icon or bitmap quickly

Checking and unchecking check boxes quickly

Enabling or disabling controls quickly

Setting or clearing radio buttons quickly

Setting or getting the caption of a button quickly

Setting or getting the focus quickly

Setting or getting the selection in a combo box quickly

Setting or getting the selection in a list box quickly

Setting or getting the text of a field quickly

Using dialog box buttons to display additional options

Validating an entry in a dialog box

See also:

Determining what the user enters in an interactive dialog box
Displaying a dialog box from within a book
Responding to a specific action
Responding to Escape and Enter keys
TB30DLG.DLL functions
Windows notification messages

Adding or deleting items in a combo box quickly

Step-by-step

SetComboBoxItems( ) sets the text for an item or items in the drop-down list of a combo box control.

AddComboBoxItem( ) adds an item to the drop-down list of a combo box control.

DeleteComboBoxItem( ) deletes an item from the drop-down list of a combo box control.

DeletenComboBoxItem( ) deletes an item from the drop-down list of a combo box control based on its index number in the list (for example, it deletes the nth item in the list).

Call the function once for each item you need to add or delete.

setComboboxItems( )

```
to get TBKDialogInit hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Initialize combo box items with selections from a field
  --on page "Setup"

  get setComboboxItems(hDlg,"Combobox Titles",text of field "Titles"\ of
page "setup")
end
```

addComboboxItem( )

deleteComboboxItem( )

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Add a new item into the sorted position in the list of combo box  --
items and remove the oldTitle from the list of drop-down items
  if update = 1
    index = -1
    sortFlag = 1
    get addComboboxItem(hDlg,"Combobox Titles","OS Manual", index,\
  softFlag)
    get DeleteComboboxItem(hDlg,"Combobox Titles","User Manual")
  end
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Setting or getting the selection in a combo box quickly

Step-by-step

The `getControlText( )` function returns the text of the combo box's edit field, which usually represents the selected item.

The `setControlText( )` function sets the corresponding text.

```
to get TBKDialogInit hDlg, hWndFocus
  get setControlText(hDlg, "combobox Destination","Tulsa")
  return 0
end get

to get TBKDialogCommand hDlg, ctrlID, hWndCtrl, message, ctrlName
  conditions
  when getControlText(hDlg,"combobox Destination") = "Austin"
    request "Austin Texas"
    return 0
  else
    return 1
  end conditions
end get
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Adding or deleting items in a list box quickly

Step-by-step

The addListBoxItem( ) function adds an item to a list box control.

The deleteListBoxItem( ) function deletes an item from a list box control.

The deletenListBoxItem( ) function deletes an item from a list box control based on its index number in the list (for example, it deletes the nth item in the list).

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --If an item doesn't exist in the list box, add it in its sorted    --
position and remove the old item
  index = -1
  softFlag = 1
  newItem = "Setting"
  oldDelete = "Spiking"
  CurrentItems = getListBoxItems(hDlg,"Listbox VolleyballSkills")
  if newItem is not in CurrentItems
    get addListBoxItem(hDlg,"Listbox VolleyballSkills",newItem,\
  index,sortFlag)
  end
  get deleteListBoxItem(hDlg,"Listbox VolleyballSkills",oldItem)
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Setting or getting the selection in a list box quickly

Step-by-step

The `getListBoxSelection( )` function returns the text of the selected line or lines in a list box control.

The `getnListBoxSelection( )` function returns the index number or numbers of selected lines in a list box.

The `setListBoxSelection( )` function sets the selected line or lines in a list box.

The `setnListBoxSelection( )` function sets the selected line or lines in a list box by reference to their index numbers (for example, it selects the nth item in the list).

getListboxSelection( )

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Get the text of the selected line of a list box
  SelectedPlayer =  getListBoxSelection(hDlg,"Listbox PlayerNames")
end
```

getnListboxSelection( )

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Get the textline number of the selection of a listbox
  SelectedPlayersPosition =  getnListBoxSelection(hDlg,\
  "Listbox PlayerNames")
end
```

setListboxSelection( )

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Highlight the appropriate selection of a list box entered by a    --
user
  get setListboxSelection (hDlg,"Listbox PlayerNames","ToolBook")
end
```

setnListboxSelection( )

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Highlight the textlines of a listbox
  textlineNumbers = "1,3,5,7"
  get setnListboxSelection (hDlg,"Listbox PlayerNames",\
  textlineNumbers)
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Changing an icon or bitmap quickly

Step-by-step

The `setBitmapData()` function identifies a bitmap resource for use in a bitmap control.

The `setIconData()` function identifies an icon resource for use in an icon control.

`setBitmapData()`

```
to get TBKDialogInit hDlg, CtrlID, hWndCtrl, message, ctrlName
  conditions
    when pageNumber = 1
      bitmapName = "bitmap" & pageNumber
      hBitmap = GDIHandle(bitmapName)
      get setBitmapData(hDlg,"BITMAP BitmapBox",hBitmap)
    end
  end
end
```

`setIconData()`

```
to get TBKDialogInit hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Randomize the icon displayed in a dialog box
  IconNum = random(6)
  Iconname = "icon" & IconNum
  hIcon = GDIHandle(Icon IconName)
  get setIconData(hDlg,"ICON IconBox",hIcon)
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback() functions

Checking and unchecking check boxes quickly
Setting or clearing radio buttons quickly

Step-by-step

The `isButtonChecked( )` function tells you if a check box is checked or a radio button is selected.

The `setButtonCheck( )` function checks or unchecks a check box control or selects or deselects a radio button control.

If a radio button is selected and you select another one, the first one is deselected.

The third parameter changes the control. If it is 0, the checkmark is removed or the control is deselected; if it is 1, the checkmark is added or the control is selected.

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --If the button "SystemDefaults" is checked then check
  --the buttons TabWidth8 and SaveOnClose but uncheck
  --the button "Display Rulers"
  if isButtonChecked (hDlg,"Button SystemDefaults") = 1
    get setButtonCheck(hDlg,"Button TabWidth8",1)
    get setButtonCheck(hDlg,"Button SaveOnClose",1)
    get setButtonCheck(hDlg,"Button DisplayRulers",0)
  end
end


----------
to get TBKDialogCommand hDlg, ctrlID, hWndCtrl, message, ctrlName
  conditions
  when ctrlName = "button choice" and message = 0
    get setButtonCheck(hDlg, "button option3", 1)
    return 1
  end conditions
end get
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Enabling or disabling controls quickly

Step-by-step

The `enableControl( )` function enables or disables a control.

The `isControlEnabled( )` function returns 1 if the specified control is enabled; 0 if it is disabled.

The third parameter, `<enable>`, enables or disables the control. If the parameter is 0, the control is disabled; if it is 1, the control is enabled.

`enableControl( )`

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  conditions
    when ctrlName = "EDIT Notes"
      if getControlText(hDlg,ctrlName) = null
        get enableControl (hDlg,"Button AddText",0)
          else
        get enableControl (hDlg,"Button AddText",1)
      end
  end
end
```

`isControlEnabled( )`

```
to get TBKDialogInit hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Make sure upon showing dialog box that proper controls
  -- are enabled; enable the init button and disable the exit
  -- button
  if isControlEnabled(hDlg,"Button Init") = 0
    get enableControl(hDlg,"Button Init",1)
    get enableControl(hDlg,"Button Exit",0)
  end
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Setting or getting the caption of a button quickly
Setting or getting the text of a field quickly

Step-by-step

The `getControlText( )` function returns the text of the specified control. For buttons, the function returns the text of the button's caption. For combo boxes, the function returns the text of the combo box's edit field, which usually represents the selected item. For fields, the function returns the text in the field.

The setControlText( ) function sets the corresponding text.

```
to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Get the name of the file that the user typed into the
  --control EDIT FileField
  FileToFind = getControlText (hDlg,"EDIT FileField")
  if FileToFind = "rescue.doc"
    get setControlText(hDlg,"BUTTON Topics","Emergency Procedures")
  end
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Setting or getting the focus quickly

Step-by-step

The `getDialogFocus( )` function returns the name of the control that has the input focus in the dialog box.

The `setDialogFocus( )` function sets the input focus to the specified control.

`getDialogFocus( )`

```
to get TBKDialogInit hDlg, CtrlID, hWndCtrl, message, ctrlName
  -- Prevent users from entering first names if they haven't
  -- entered last names
  if getDialogFocus(hDlg,"EDIT Firstname") = 1 and \
     getControlText(hDlg,"EDIT Lastname") = null
    request "Please enter last name"
  end
end
```

`setDialogFocus( )`

```
to get TBKDialogInit hDlg, CtrlID, hWndCtrl, message, ctrlName
  --Have the dialog box come up with a certain field always having
--the focus first
  get setDialogFocus(hDlg,"edit Lastname")
end
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Using dialog box buttons to display additional options

Step-by-step

```
to get TBKDialogCommand hDlg, ctrlID, hWndCtrl, message, ctrlName
  conditions
  when ctrlName = "delete" and message = 0
    request "Are you sure you want to delete the item?"
    return 0
  else
    return 1
  end conditions
end get
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Validating an entry in a dialog box

Step-by-step

```
to get TBKDialogCommand hDlg, ctrlID, hWndCtrl, message, ctrlName
  conditions
  when getControlText(hDlg,"title") <> "supervisor"
    request "You must be a supervisor to use this function."
    get endTBKDialog(hDlg,1)
    return 0
  else
    return 1
  end conditions
end get
```

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions

Windows notification messages

This table shows the Windows notification messages, their numbers, and their meanings.

Windows message	Message number Hex	Decimal	Meaning
BN_CLICKED	0x0000	0	The user clicked a button.
BN_DISABLE	0x0004	4	A button is disabled.
BN_DOUBLECLICKED	0x0005	5	The user double-clicked a button.
BN_HILITE	0x0002	2	The user highlighted a button.
BN_PAINT	0x0001	1	The button should be painted.
BN_UNHILITE	0x0003	3	The highlight should be removed.
CBN_CLOSEUP	0x0008	8	The drop-down list of a combo box has closed.
CBN_DBLCLK	0x0002	2	The user double-clicked a string.
CBN_DROPDOWN	0x0007	7	The drop-down list of a combo box is dropping down.
CBN_EDITCHANGE	0x0005	5	The user has changed text in the edit control.
CBN_EDITUPDATE	0x0006	6	Altered text is about to be displayed.
CBN_ERRSPACE	0xFFFF	65535	The combo box is out of memory.
CBN_KILLFOCUS	0x0004	4	The combo box is losing the input focus.
CBN_SELCHANGE	0x0001	1	A new combo box list item is selected.
CBN_SELENDCANCEL	0x000A	10	The user's selection should be canceled.
CBN_SELENDOK	0x0009	9	The user's selection is valid.
CBN_SETFOCUS	0x0003	3	The combo box is receiving the input focus.
EN_CHANGE	0x0300	768	The display is updated after text changes.
EN_ERRSPACE	0x0500	1280	The edit control is out of memory.
EN_HSCROLL	0x0601	1537	The user clicked the scroll bar.
EN_KILLFOCUS	0x0200	512	The edit control is losing the input focus.
EN_MASTEXT	0x0501	1281	The insertion is truncated.
EN_SETFOCUS	0x0100	256	The edit control is receiving the input focus.
EN_UPDATE	0x0400	1024	The edit control is about to display altered text.
EN_VSCROLL	0x0602	1538	The user clicked the vertical scroll bar.
LBN_DBLCLK	0x0002	2	The user double-clicked a string.
LBN_ERRSPACE	0xFFFE	65534	The list box is out of memory.
LBN_KILLFOCUS	0x0005	5	The list box is losing the input focus.
LBN_SELCANCEL	0x0003	3	The selection is canceled.
LBN_SELCHANGE	0x0001	1	The selection is about to change.
LBN_SETFOCUS	0x0004	4	The list box is receiving the input focus.

See also:
Displaying a dialog box from within a book
Using dialogCallback( ) functions

TBKDialogInit

Syntax `TBKDialogInit(<hDlg>, <hWndFucus>)`

Declaration `to get TBKDialogInit hDlg, hWndFocus`

Parameters

`<hDlg>`	Window handle of the dialog box.
`<hWndFocus>`	Window handle of control with the focus.

Description OpenScript `to get` handler that gets evaluated on WM_INITDIALOG

Returns

0	No default processing.
1	Default processing should occur.

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions
Windows notification messages

TBKDialogCommand

Syntax `TBKDialogCommand(<hDlg>, <control ID>, <control window handle>, <notification message>, <control name>)`

Declaration `to get TBKDialogCommand hDlg, CtrlID, hWndCtrl, Msg, CtrlName`

Parameters

`<hDlg>`	Window handle of the dialog box.
`<control ID>`	ID of control sending notification message.
`<control window handle>`	Window handle of control sending notification message.
`<notification message>`	Notification message number.
`<control name>`	NAME of control sending notification message.

Description OpenScript `to get` handler that gets evaluated on WM_COMMAND

Returns

0	No default processing.
1	Default processing should occur.

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions
Windows notification messages

TBKDialogDestroy

Syntax `TBKDialogDestroy(<hDlg>)`

Declaration `to get TBKDialogDestroy hDlg`

Parameters `<hDlg>` Window handle of the dialog box.

Description OpenScript `to get` handler that gets evaluated on WM_DESTROY

Returns

0	No default processing.
1	Default processing should occur.

See also:

Displaying a dialog box from within a book
Using dialogCallback( ) functions
Windows notification messages

TB30DLG.DLL

The Windows dialog box DLL supplied with ToolBook. It contains many functions you can use to display and interact with dialog boxes created using the Dialog editor.

.DIA file

A text file that contains the dialog box template and the initial values for the dialog box controls.

Responding to Escape and Enter keys

Step-by-step

When the Escape or Enter key is pressed, a message is sent to the `TBKDialogCommand` handler. The <`CtrlID`> parameter of the message is 1 if the Enter key was pressed and 2 if the Escape key was pressed.

If the Enter key was pressed and there is a pushbutton control on the dialog box with an ID of OK, the <`hwndCtrl`> and <`controlName`> parameters of the message are set as if that pushbutton had been clicked.

If the Escape key was pressed and there is a pushbutton control on the dialog box with an ID of Cancel, the <`hwndCtrl`> and <`controlName`> parameters of the message are set as if that pushbutton had been clicked.

If there is no control whose ID corresponds with the <`CtrlID`> parameter of the message, the <`hwndCtrl`> and <`controlName`> parameters of the message are set to 0 and " ", respectively.