Asymetrix Multimedia ToolBook 3.0

Release Notes

July 11, 1994

This document contains the same information available in the Release Notes online Help application (RELNOTES.HLP). Use this version if you want to print portions of the information or copy and paste code examples into your applications.

Contents

1.0. Features new in Multimedia ToolBook 3.0

Multimedia ToolBook 3.0 contains hundreds of new features, including a new interface, new and enhanced objects, and many improvements to OpenScript.

1.1. New interface tools

* **Tool bar**. Execute commands directly by clicking buttons on Multimedia ToolBook's new tool bar.

* **Ruler shadows**. When you select an object, the object's dimensions display as shadows in the ruler so you can more precisely size and move it.

* **Status bar**. Display your own messages in the status bar. Turn the status bar on and off for any viewer in your application.

* **Right-click menus**. Click the right mouse button to display a context-sensitive menu and tool bar you can use to manipulate an object as well as view and edit its properties.

* **New Tools menu**. Choose from a wide range of custom tools that make authoring easier and more efficient, including a Properties Browser, a window to define startup preferences, and other timesaving options.

* **Command window history**. Recall previously executed commands in the Command window using the Command window history.

* **Recently used books on File menu**. Open books quickly by choosing from the four most recently used books listed in the File menu.

* **Duplicate menu item**. Choose Duplicate from the Edit menu or press Ctrl+D to duplicate the selected object.

See also
ToolBook User Manual, Chapter 2, "ToolBook orientation"
ToolBook User Manual, Chapter 20, "Using the scripting tools"

1.2. Tools menu

Adding MTB30.SBK to the `sysBooks` property adds a Tools menu at Author level that contains useful authoring tools, including utilities to:

* view and change an object's properties or any system variables.
* set startup preferences.
* find and replace text in all scripts in the book.
* help you align objects, add three-dimensional effects, and set layer order.
* import and export text to and from fields and record fields.
* create hyperlinks between pages of your book.
* draw path animations.
* copy stages with prescripted controls into your book to add digital video without programming.

For details about the Tools menu, refer to the Menu commands in the Multimedia ToolBook online Help.

1.3. Enhanced text support

* **Full-text search.** Create an index of all the words in your application and search it to quickly locate any page in your application.

* **Check spelling.** Polish your application by checking the spelling of words in any object that contains text.

* **Embed TrueType fonts**. Include with your application the fonts you used to create it, guaranteeing that the application will look the same on every system.

Other enhancements to text in Multimedia ToolBook are listed under "Fields and record fields."

1.4. Improved importing

* **Importing pages**. Import pages from other books in one step. Multimedia ToolBook automatically imports the associated resources and remaps the record field text.
* **Importing graphics**. Import a wider variety of graphics.

See also
ToolBook User Manual, Chapter 16, "Importing and exporting data"

1.5. Printing improvements

* **Updated interface**. Use the updated printing user interface for easier access to printing.

* **WYSIWYG word wrapping**. Design your printed output more accurately with WYSIWYG (what you see is what you get) word wrapping, which makes text look the same on the screen as it does on the printed page.

* **Printing colors**. Control how well colors are rendered on black-and-white printers.

* **Page scaling**. Create more flexible printer output with improved page scaling.

* **Robust printing**. Enjoy consistent and reliable output with more robust printing support.

See also
ToolBook User Manual, Chapter 15, "Printing from ToolBook"

1.6. Enhanced color support

* **Solid colors**. Display custom solid object colors by setting a book's `solidColorsEnabled` property to `true`.

* **RGB values**. Set colors using RGB values instead of, or in addition to, HLS values.

* **New constants**. Use the new color constants `gray` and `lightGray`.

* **Palette shift**. Eliminate palette shift by assigning a shared color palette to a book.

* **16- and 24-bit color support**. Take advantage of 16- and 24-bit display devices in your applications.

* **Graphics of all color depths**. Add 8-bit, 16-bit, or 24-bit graphics to your application. Multimedia ToolBook 3.0 automatically dithers the images to the color depth of the user's machine.

* **Windows colors**. Set objects to use the colors set in the Windows Control Panel with the new `useWindowsColors` property.

See also
ToolBook User Manual, Chapter 3, "Drawing objects"
ToolBook User Manual, Chapter 23, "Variables, values, and text"
ToolBook User Manual, Appendix B, "Working with 256-color display devices"

1.7. Managing embedded objects as resources

* **Centralized resources**. Maintain a centralized database of resources, including menu bars, icons, cursors, bitmaps, and color palettes, that you can use throughout your book.

* **Menu bars**. Create and modify menu bars without scripting using the new Menu Bar editor.

* **Icons and cursors**. Create and edit icons and cursors using the new Icon/Cursor editor.

* **Bitmaps and color palettes**. Create and edit bitmaps and color palettes, including those created with other Windows programs, using BitEdit or PalEdit.

* **Wave audio files**. Edit wave audio files (.WAV files) using WaveEdit.

* **Single-source changes**. Update every occurrence in a book of a bitmap, icon, color palette, or cursor with a single change. For example, if several graphic buttons share the same bitmap resource, Multimedia ToolBook automatically updates each button when you edit or replace the bitmap.

* **Clip art**. Get a head start on user interface design by importing predefined icons, cursors, and bitmaps from the library of clip art included with Multimedia ToolBook 3.0.

See also
ToolBook User Manual, Chapter 14, "Using Windows resources"

1.8. Optimizing for CD-ROM

* **Optimized format**. Store your application in a format optimized for access on a CD-ROM.

* **Cache file**. Create a cache file containing frequently-used information to speed access to CD-ROM-based applications.

1.9. New tools for distributing applications

* **.EXE files**. Save books as .EXE files that you can launch directly from the Program Manager without starting Multimedia ToolBook or Runtime Multimedia ToolBook first. (Multimedia ToolBook must still be available in your path, however.)

* **CD-ROM optimization**. Save your application in a format optimized for access on CD-ROM. Create a temporary or permanent cache file to speed access to CD-ROM-based applications.

* **Installation utility**. Package your applications with the new Asymetrix Setup Utility, which allows you to create installable components, assign icons to each file, and compress files for distribution on floppy disks or CD.

See also
ToolBook User Manual, Chapter 17, "Preparing applications for release"

1.10. Multimedia clips

To access multimedia in Multimedia ToolBook you create clips, which are references to a segment of a media file (such as an .AVI file) or a media device (such as a CD in a CD-ROM player). A clip includes these attributes:

* A name that you assign

* A start point

* An end point

Depending on the type of clip, it might also include additional information. For example, a clip referencing a digital audio file might include information about how loud the clip should play.

Creating clips allows you to:

* keep information about the multimedia elements of your application in a central location to simplify application development and maintenance.

* create search paths for the clips to make your application path-independent.

* edit attributes of the media clip (such as changing its start and end points) without changing scripts that reference the clip.

* incorporate still images (.BMP, .DIB, .WMF, .GIF, PICT, and Kodak Photo CD) into your application using the same techniques used for video types.

You can create clips for any MCI-compatible media type, including these:

Type	**Example**
Video	Video For Windows, QuickTime, videodisc, video overlay, and VISCA-compatible VCRs
Audio	Wave audio, Musical Instrument Digital Interface (MIDI), CD audio
Animation	Macromedia Director movies, Autodesk® .FLI and .FLC files

1.11. Stages

To display visual media such as digital video files, you can create a stage, which is an object that defines a position and size for the clip. Multimedia stages offer these advantages:

* **Position and size visual media**. Draw a stage to set the location where visual media such as digital videos will play and to adjust the size of the play window.

* **Device independence**. Eliminate device-dependency problems because the stage is automatically resized if you change video devices.

* **Prescripted objects**. Use the Media Widgets tool to drag and drop prescripted stages into your application. These stages contain controls to play, stop, and pause visual media.

* **Borders**. Add beveled borders to the stage for visual appeal.

* **Transitions**. Add transition effects (wipes, fades, slides) to the stage that play at the beginning or end of a clip.

1.12. OpenScript support for multimedia

Using new OpenScript commands, you can

* play, pause, stop, and rewind multimedia.

* get and set properties of clips (references to media files) such as their start and end points, volume, and display size.

* create new clips or import multimedia elements.

* adjust the volume of MIDI, wave audio, and CD audio.

Multimedia ToolBook also includes commands to execute MCI commands directly to take advantage of device-specific features.

1.13. Media packager

To help you manage clips efficiently, Multimedia ToolBook includes a Media Packager utility, which

* removes explicit path references from clips.

* enables you to set up media search paths that Multimedia ToolBook uses to locate a clip.

* enables you to move or copy media source files.

For details about the Media Packager utility, refer to the Multimedia ToolBook 3.0 online Help.

1.14. Clip media

Multimedia ToolBook includes a wide selection of sound, video, and still-image clip media that you can use in your applications. Clip media files are located in directories below the \MEDIA directory on the

Multimedia ToolBook 3.0 CD.

Note You have limited distribution rights to some of the clip media included with Multimedia ToolBook 3.0. For details about distributing clip media with your application, refer to the Multimedia ToolBook 3.0 license agreement.

1.15. Digital Video Producer

Included with Multimedia ToolBook 3.0 is Digital Video Producer, a standalone utility for working with digital video files (.AVI files). Digital Video Producer enables you to create and edit digital videos, and add effects such as transitions and titles.

For information about Digital Video Producer, click its icon in the Multimedia ToolBook 3.0 Program Manager group, then choose Help or press F1.

1.16. Drag and drop

* **Drag objects**. Define drag-and-drop behavior for any object to create highly interactive and intuitive applications.

* **Drag-and-drop properties**. Specify drag-and-drop properties using dialog boxes or create conditional drag-and-drop behavior in OpenScript.

* **Drag and no-drop cursors**. Use any graphic resource as a drag cursor, including full-color bitmaps (a capability unique to ToolBook).

See also
ToolBook User Manual, Chapter 27, "Using special effects"

1.17. Path animation

* **Create animations without programming**. Draw animation paths for any object by pointing and clicking with the mouse. Set the duration and speed of the animation using a dialog box.

* **Create cel animations.** Create animations that show one view of an object after another (such as a series of bitmaps of a globe to suggest the spinning of the earth) by grouping objects and setting options in a dialog box.

For details about path animation, refer to the Path Animation online Help system provided separately with Multimedia ToolBook.

1.18. Buttons

* **3D buttons**. Create three-dimensional radio buttons and check boxes for your application interface.

* **Graphical buttons**. Create custom checkboxes, radio buttons, and pushbuttons by adding graphics to your buttons.
* **Radio buttons**. Create radio button groups more easily with the new `autoRadioButtons` group property.
* **Disabled buttons**. Prevent users from using buttons by disabling the buttons with the new `enabled` property.
* **Mnemonic access characters**. Define custom keyboard interfaces in your application by assigning mnemonic access characters to your buttons, so that users can use Alt plus a keystroke to click the button.

See also
ToolBook User Manual, Chapter 5, "Adding buttons"

1.19. Fields and record fields

* **Inline graphics**. Paste bitmaps and icon graphics into a field, or insert a graphic that you've imported into Multimedia ToolBook's resource system.
* **Borderless fields**. Create transparent, borderless fields in one step using the borderless field tool.
* **Label buttons**. Create labels next to fields or in group boxes to define mnemonic access characters for objects that don't have captions.
* **Superscripts and subscripts**. Add superscripts and subscripts to your field text (invaluable for displaying mathematical and scientific data).
* **Color text**. Add visual emphasis to your fields with color. Each character in a field can be a different color.
* **Inset and raised styles.** Create three-dimensional fields and record fields using new styles.
* **Rich-text format (RTF) support**. Paste formatted text from word processing programs into fields or import it directly from a DOS file. Reduce development of text-rich applications by eliminating time spent reformatting imported text. Move formatted text, including hotwords, from one field to another using the `richText` field property.

See also
ToolBook User Manual, Chapter 6, "Adding text"
ToolBook User Manual, Chapter 7, "Adding record fields"

1.20. Groups

* **Group editor**. Select objects in a group individually so you can edit them without destroying the group. You can also add objects to groups by selecting an object in the group and then drawing the new object, which automatically becomes part of the group.
* **Add or delete objects**. Add or delete objects in existing groups.

* **Auto-radio buttons**. Define radio button behavior more easily with the new `autoRadioButtons` property.

See also
ToolBook User Manual, Chapter 3, "Drawing objects"

1.21. Paint objects (bitmaps)

* **Transparent bitmaps**. Create paint objects with "see-through" sections that allow you to see the objects underneath the bitmap. For details, refer to the Multimedia ToolBook online Help.

1.22. Hotwords

* **Hotword styles**. Define hotwords in color, as dotted or as underlined (which matches the styles available in Windows Help). Define a single hotword style for an entire book for convenience, but override it for individual hotwords for flexibility. For details about new hotword styles, refer to the Multimedia ToolBook online Help.
* **Importing hotwords**. Define hotwords in your word processing program, then import them into your Multimedia ToolBook fields. This feature is very useful to anyone who is creating hypertext applications in Multimedia ToolBook.
* **Finer control**. Control hotwords more precisely with the new `text`, `textOffset`, and `bounds` hotword properties.
* **Hotword graphics**. Create hotword graphics by including graphics in your hotwords.

See also
ToolBook User Manual, Chapter 9, "Adding hotwords"

1.23. Shapes and lines

* **Reshape objects**. Reshape polygons and arcs more easily with an enhanced Reshape command that allows you to add and remove vertices.
* **Line ends**. Add arrowheads and other effects to lines with the new line ends palette.

See also
ToolBook User Manual, Chapter 3, "Drawing objects"

1.24. Menus

* **Menu Bar editor**. Use the new Menu Bar editor to create and edit menu resources without programming. You can define multiple levels of cascading menus. Multimedia ToolBook 3.0

comes with a predefined Reader-level menu resource.

Use menu resources instead of modifying menus in the `enterBook` handler, as you did in Multimedia ToolBook 1.5.

* **`menuItem` selected message**. Write a handler for the new `menuItemSelected` message instead of writing individual handlers for each menu item. (Multimedia ToolBook still sends the menu command and alias messages if there is no `menuItemSelected` handler.)

* **New OpenScript commands**. Control menus more precisely with new OpenScript commands and functions that support menus, such as `menuEnabled`, `setMenuName`, `setMenuHelpText`, `removeSeparator`, and others.

* **Popup menus**. Display popup menus with the new `popupMenu()` function.

Tip You can use the tool bar to send many of the same commands that are defined in menus.

See also
ToolBook User Manual, Chapter 13, "Creating and modifying menu bars"

1.25. Viewers (multiple windows)

* **Multi-window applications**. Display pages and create multi-window applications, dialog boxes, palettes, tool and status bars, popup windows, nested child windows, and much more to display your information in a variety of ways.

* **Multiple concurrent windows**. Display multiple pages simultaneously.

* **Dialog boxes and palettes**. Create deluxe dialog boxes and palettes that make use of all of Multimedia ToolBook's color and graphics capabilities.

* **Popup windows**. Create popup viewers that disappear when you click the mouse.

* **Captions**. Add thin captions to create low-profile windows such as tool palettes.

* **Replace multiple instances**. Combine multiple Multimedia ToolBook instances in a single, powerful, easily-managed application.

See also
ToolBook User Manual, Chapter 11, "Creating windows with viewers"

1.26. Combo boxes

* **Drop-down list box**. Provide the choices of a list box in the space of a single-line field by creating combo box objects with drop-down lists.

* **Sort items**. Sort items in the drop-down list automatically.

See also
ToolBook User Manual, Chapter 8, "Creating list boxes and combo boxes"

1.27. OLE objects

* **Embed OLE 1 objects**. Embed or link any OLE 1.0 object into your Multimedia ToolBook application.

* **Embed data from other Windows programs**. Add graphs, drawings, and other objects to your Multimedia ToolBook pages by pasting them from OLE-compatible applications.

See also
ToolBook User Manual, Chapter 12, "Using object linking and embedding"

1.28. Improved Script editor

* **Multiple windows**. Edit multiple scripts simultaneously in multiple modeless Script editors.

* **Tool bar**. Use the Script editor tool bar as a shortcut for most commands.

* **Exchange text**. Import and export text files directly into and from the Script editor.

* **Block commenting**. Comment and uncomment blocks of code in one step.

* **Block indenting**. Promote and demote blocks in one step.

* **Parent script editing**. Edit the scripts of parent objects by choosing the name of the parent object from a menu.

See also
ToolBook User Manual, Chapter 20, "Using the scripting tools"

1.29. Improved Debugger

* **Edit variables**. View and edit variable values of any length.

* **Command window**. Inspect and change any property using the Command window while in the Debugger.

See also
ToolBook User Manual, Chapter 20, "Using the scripting tools"

1.30. Auto-Script script library utility

* **Learning tool**. Learn Multimedia ToolBook quickly by browsing through predefined handlers, inserting a handler into a script, and testing the results.

* **Copy scripts**. Insert predefined handlers from a script library using the Auto-Script feature. You can customize options in each handler you insert.

* **Edit existing scripts**. Add your favorite handlers to an Auto-Script library file.

* **Programmer productivity**. Share Auto-Script libraries among multiple programmers.

See also
ToolBook User Manual, Chapter 20, "Using the scripting tools"

1.31. Improved Command window

* **Command window history**. The Command window maintains a history of the most recent commands you executed. Use the PageUp and PageDown keys to review and recall the commands or execute any command by double-clicking it.

* **Support for Ctrl+arrow keys**. Jump from word to word using Ctrl+Left Arrow or Ctrl+Right Arrow.

* **Properties stored between sessions**. Command window position, size, and split bar height are written to MTB30.INI when you close the program. They're restored when you start Multimedia ToolBook again and display the Command window.

Note You can no longer evaluate expressions directly in the Command window.

See also
ToolBook User Manual, Chapter 20, "Using the scripting tools"

1.32. Improved OpenScript performance and efficiency

* **Faster compiler**. Write scripts that are up to 10 times faster than scripts in Multimedia ToolBook 1.5 with the new optimizing OpenScript compiler. Looping and access to program variables are key areas where performance has been improved.

* **Larger scripts**. Write more complex programs that use larger variables. Each variable in OpenScript can contain 64K of data, and you can have up to 16MB of variable data (an increase from 32K in Multimedia ToolBook 1.5).

* **Optional typed variables**. Improve efficiency of your programs by using optional typed variables, including types such as `int`, `word`, `string`, `object`, and many others.

1.33. Improvements to variables

* **Optional typed variables.** Improve efficiency of your programs by using optional typed variables, including types such as `int`, `word`, `string`, `object`, and many others.

* **Larger variables.** Write more complex programs that use larger variables; each variable in OpenScript can contain 64K of data, and you can have up to 16MB of variable data -- an increase from 32K in Multimedia ToolBook 1.5.

* **New assignment operator.** Assign values to variables and properties more conveniently using the new equal assignment operator (=), which also makes your OpenScript handlers faster to type and easier to read.

* **Arrays.** Access structured data quickly and easily using arrays of up to 16 dimensions. Use fixed arrays for fast, efficient management of data and dynamic arrays to store variable data whose size changes dynamically. Each array element can contain up to 65,536 characters.

See also
ToolBook User Manual, Chapter 23, "Variables, values, and text"

1.34. New operators

* **Assignment operator**. Assign values to variables and properties more easily using the new equal assignment operator (=), which also makes your OpenScript handlers simpler to read.

* **Embedded graphics**. Manipulate graphics embedded in text with the new `graphic` text operator.

* **Bitwise operators**. Manipulate individual bits in numbers using the new bitwise operators. This capability is important when you're working with DLLs and certain algorithms.

* **Variable names**. Use the at symbol (@) to distinguish variable names from OpenScript keywords or property names.

* **Variable as message**. Use parentheses with the `send` command to evaluate an expression as a message.

See also
ToolBook User Manual, Chapter 22, "Statements, functions, and expressions"

1.35. Improvements to messages

* **Clicking mouse buttons**. Make buttons and other objects respond more like standard Windows controls with the new `buttonClick` message -- the message is only sent if the user releases the mouse button while the cursor is still over the object.

* **Linking and unlinking system books**. Write more reliable system books with the new `linkSysBook` and `unlinkSysBook` messages. These messages notify your system book when it is time to initialize or clean itself up, so you no longer need to rely on forwarded `enterSystem`, `enterBook`, or `enterPage` messages.

* **Entering and leaving applications**. Track when a book is opened and closed in the Main window with the `enterApplication` and `leaveApplication` messages. Because you can now display multiple books concurrently using viewers, you can use these messages in place of `enterBook` and `leaveBook` to perform application initialization and cleanup.

* **Pressing Alt+key combinations**. Write handlers for the new `keyMnemonic` message to define your own Alt+key combinations.

* **Sending variables as messages**. Send the contents of variables as messages using an enhanced `send` command. For example:

```
ask "Send what message?"
send (It)
```

* **Request notification of messages**. Write handlers that are notified when messages reach the current page or background. This allows you to create self-contained objects that maintain their customized behavior when they are copied.

See also
ToolBook User Manual, Chapter 21, "Messages, objects, and properties"
ToolBook User Manual, Chapter 25, "Handling user events"

1.36. Notify handlers

* **Self-contained objects**. Build intelligent, self-contained objects using the new notify handlers, which execute in response to messages that are sent to the object's page. For example, you can write a clock object that automatically updates the current time whenever the page receives an idle message, or you can write an animation object that starts playing when the page is entered. You can copy and paste these objects onto any page and they will work with no further scripting.

* **Libraries**. Use notify handlers to create libraries of intelligent objects that you and other Multimedia ToolBook authors can reuse.

See also
ToolBook User Manual, Chapter 21, "Messages, objects, and properties"

1.37. New commands and functions (overview)

* **Page transitions**. Add a variety of special effects to page navigation using the new `transition` command. Choose from `blinds`, `drip`, `push`, `slide`, `spiral`, and many others.

* **Number formats**. Format numbers as binary or hexadecimal with new options for the `format` command.

* **Importing books**. Import other books with the `import book` command.

* **Importing pages**. Import any single page or range of pages from any other book with the `import pages` command.

* **Copying objects**. Copy objects directly, without the Clipboard, using the `copyObject()` function.

* **Verify objects**. Test for the existence of an object with the `isObject()` function.

* **Verify formats**. Test whether a value matches a format using the `isType()` function.

* **Sounds**. Play .WAV files through your sound board using the `playSound()` function.

* **Financial functions**. Create "money-smart" applications using a wide range of financial functions.

* **Converting coordinates**. Convert easily between page units and pixels using new coordinate conversion functions.

* **Popup menus**. Display popup menus using the `popupMenu()` function.

See also
ToolBook User Manual, Chapter 11, "Creating windows with viewers"
ToolBook User Manual, Chapter 13, "Creating and modifying menu bars"
ToolBook User Manual, Chapter 16, "Importing and exporting data"

1.38. New commands

Multimedia ToolBook 3.0 includes the following new OpenScript commands. For more details about each command, see the OpenScript Reference Manual or the online Help.

Multimedia commands

Command	Description
`installFontResource`	Copies a TrueType font from the book to the current computer.
`mmClose`	Unloads a media clip loaded previously.
`mmCue`	Moves the start point of a clip.
`mmHide`	Hides the current frame of a visual clip.
`mmOpen`	Loads a clip into memory and readies it for playing.
`mmPause`	Temporarily stops a clip from playing.
`mmPlay`	Begins playing a clip.
`mmRewind`	Returns to the start point of a clip.
`mmSeek`	Moves to a particular point in a clip.
`mmShow`	Shows one frame from a visual clip.
`mmStop`	Stops a clip from playing.
`mmYield`	Yields processing time so that multimedia drivers can refresh buffers and process messages (this command is documented in online Help only).
`new clip`	Creates a new clip from a media file or media device.

Menu commands

Command	Description
`disable menu`	Deactivates an entire menu.
`disable menuItem`	Deactivates a single menu item.
`enable menu`	Activates a menu previously disabled with `disable menu`.
`enable menuItem`	Activates a menu item previously disabled with `disable menuItem`.
`remove separator`	Removes the line between items in a menu.

Objects

Command	Description
`align <type>`	Aligns selected objects.
`copy resource`	Makes another copy of an embedded bitmap, icon, cursor, palette, or

	menu bar.
`drag`	Starts a drag-and-drop operation.
`sendNotifyAfter`	Sends a message that triggers an object's `notifyAfter` handler.
`sendNotifyBefore`	Sends a message that triggers an object's `notifyBefore` handler.

Resource commands

Command	Description
`export resource`	Copies an embedded bitmap, icon, cursor, palette, or menu bar to a DOS file.
`import resource`	Embeds a bitmap, icon, cursor, palette, or menu bar from a DOS file
`insert graphic`	Embeds a bitmap or icon resource into a field.
`replace resource`	Overwrites an embedded bitmap, icon, cursor, palette, or menu bar with another.
`remove resource`	Discards an embedded bitmap, icon, cursor, palette, or menu bar imported previously with `import resource`.

Book, page, and background commands

Command	Description
`import book`	Copies all pages from another book to the current book.
`import pages`	Copies selected pages from another book to the current book.
`save as EXE`	Saves the current book with an. EXE shell that starts Multimedia ToolBook automatically.
`transition`	Displays a special effect such as fade or wipe between pages.

Viewer (window) commands

Command	Description
`activate <viewer>`	Makes a particular viewer the current one.
`close <viewer>`	Closes a viewer.
`hide <viewer>`	Makes a viewer invisible but still available.
`in <viewer>`	Changes the active window temporarily in a script.
`new viewer`	Creates a new viewer.
`open <viewer>`	Initializes a viewer and makes its nonpersistent properties available.
`show <viewer>`	Displays a viewer opened previously with `openViewer`.

Other new commands

Command	Description
`fill <array>`	Fills all values in an array with a single value.
`seekFile`	Moves to a specified position in an ASCII file after opening the file with `openFile`.

1.39. New functions

Multimedia ToolBook 3.0 includes the following new OpenScript functions. For more details about each function, see the OpenScript Reference Manual or the online Help.

Multimedia functions

Function	Description
`callMCI()`	Executes an MCI command.
`imageCommand()`	Displays a graphic (bitmap or vector graphic).

`importPhoto()`	Copies an image from a Kodak Photo CD as a bitmap.
`mixerCommand()`	Controls sound level for different devices.
`timerCapability()`	Requests information about the timer device.
`timerStart()`	Accesses the Windows timer device.
`timerstop()`	Stops a timer started with `timerStart()`.

Menu-related functions

Function	**Description**
`menuEnabled()`	Verifies whether a menu is active.
`menuItemChecked()`	Verifies whether a menu item has a checkmark next to it.
`menuItemEnabled()`	Verifies whether a menu item is active.
`popupMenu()`	Displays a popup (tear-off) menu.
`setMenuHelpText()`	Sets the status bar text that appears when this menu is highlighted.
`setMenuItemHelpText()`	Sets the status bar text that appears when this menu item is highlighted.
`setMenuItemName()`	Changes the name of a menu item.
`setMenuName()`	Changes the name of a menu.

Coordinate conversion functions

Function	**Description**
`clientToPageUnits()`	Converts a location in a client window in pixels to page units.
`clientToScreen()`	Converts a location in a client window in pixels into an absolute location in the screen.
`frameToPageUnits()`	Converts a location within a window's frame in pixels into page units.
`frameToScreen()`	Converts a location within a window's frame in pixels into an absolute screen location.
`pageUnitsToClient()`	Converts a location in page units into a location in a client window in pixels.
`pageUnitsToFrame()`	Converts a location in page units into a location in a frame window in pixels.
`pageUnitsToScreen()`	Converts a location in page units into an absolute screen location in pixels.
`screenToClient()`	Converts an absolute location on the screen into a location in a client window in pixels.
`screenToFrame()`	Converts an absolute location on the screen into a location in a frame window in pixels.
`screenToPageUnits()`	Converts an absolute location on the screen into page units.

Financial functions

Function	**Description**
`annuityFactor()`	Returns a factor of the present value of an annuity.
`compoundFactor()`	Returns the future value of an interest-bearing account.
`ddb()`	Returns depreciation of an asset for a specific period.
`fv()`	Returns the future value of an annuity.
`ipmt()`	Returns the amount of interest to be paid on an investment.
`irr()`	Returns the interest rate for a series of cash flow amounts.
`nper()`	Returns the number of periods required for an investment.
`npv()`	Returns the present value of an investment based on cash flow amounts.
`pmt()`	Returns the periodic payment for an annuity.
`ppmt()`	Returns the payment on principal for an investment.

`pv()`	Returns the present value of an investment.
`rate()`	Returns the interest rate per period for an investment.

Resource functions

Function	Description
`chooseResource()`	Displays the Choose Resource dialog box from which you can select a bitmap, icon, cursor, color palette, or menu bar resource.
`GDIHandle()`	Returns the handle of a resource in a format that can be passed to Windows.
`resourceCount()`	Returns the number of times a resource is referenced in the book.
`resourceHandle()`	Returns the handle of a resource in a format that can be passed to Multimedia ToolBook.
`resourceList()`	Returns a list of the resource references for a specified resource type.

Full-text search functions (creating an index)

Function	Description
`ftsAddContext()`	Adds a new context to the index.
`ftsAddPage()`	Adds a page to the index.
`ftsAddSectionsToPage()`	Assigns one or more sections to a page.
`ftsAddTextToSection()`	Adds the text of a specified object to a section.
`ftsBuildIndex()`	Builds the index based on an index settings file.
`ftsInitIndex()`	Initializes an index in preparation of adding information into it.
`ftsRemoveIndex()`	Deletes an index.

Full-text search functions (searching)

Function	Description
`ftsAllContextTitles()`	Returns a list of all page contexts.
`ftsAllMatchingRefs()`	Returns all page titles from the most recent query.
`ftsAllMatchingTitles()`	Returns all page references from the most recent query.
`ftsCloseIndex()`	Closes an open index.
`ftsContextCount()`	Returns the number of contexts.
`ftsContextTitle()`	Returns the text of a context based on a context number.
`ftsGetErrorNotify()`	Returns error notification status.
`ftsGetErrorNumber()`	Returns number of most recent error.
`ftsGetErrorString()`	Returns text of most recent error.
`ftsGetOffsets()`	Returns a list of coordinates identifying the search term in a block of text.
`ftsMatchRef()`	Returns one page reference from the most recent query.
`ftsMatchTitle()`	Returns one page title from the most recent query.
`ftsNamedSections()`	Returns a list of all named sections.
`ftsOpenIndex()`	Makes an index available for searching.
`ftsQuery()`	Searches the index.
`ftsReQuery()`	Searches the index, limiting the search to the pages found in a previous query.
`ftsSetContextScope()`	Narrows a search to a specific page context.
`ftsSetErrorNotify()`	Specifies whether errors are displayed.
`ftsTagEntries()`	Returns a list of all phrases stored in a hotword tag.
`ftsTags()`	Returns a list of all hotword tags.

Other functions

Function	Description
`clipboardFormats()`	Returns a list of formats currently on the Clipboard.
`dimensions()`	Returns the dimensions of the specified array.
`flushMessageQueue()`	Clears pending keystrokes, mouse clicks, or messages.
`isObject()`	Determines if the specified object exists.
`isType()`	Determines if the specified variable matches a particular data type.
`playSound()`	Plays a .WAV file.
`windowFromPoint()`	Returns a reference to the topmost viewer (window) displayed at the specified location.
`windowRefFromHandle()`	Returns a reference to the viewer (window) identified by the specified handle.

1.40. New messages

Multimedia ToolBook 3.0 includes the following new OpenScript messages. For more details about each message, see the OpenScript Reference Manual or the online Help.

Multimedia messages

Message	Description
`imageNotify`	Sent to the specified object after an `imageCommand()` function executes.
`MCINotify`	Sent to the specified object after a `callMCI()` function executes.
`MCISignal`	Sent to the specified object by the `MCI` signal command.
`mmNotify`	Sent to the specified object after a media command such as `mmPlay` or `mmStop` executes.
`timerNotify`	Sent to the specified object after a `timerStart()` function executes.

Menu event messages

Message	Description
`align<type>`	Aligns selected objects.
`color`	Toggles display of the Color Tray.
`find`	Displays the Find dialog box.
`graphic`	If the focus is in a field and a character is selected, displays the Graphic dialog box, from which the user can choose a graphic to insert into text.
`insertGraphic`	Inserts a graphic resource (bitmap, icon) into a field.
`line`	Displays the line palette.
`lineEnds`	Displays the line ends palette.
`normalScript`	Removes superscripting or subscripting from the selected text.
`pasteSpecial`	Displays the Paste Special dialog box, from which the user can choose the format for pasting an object from the Clipboard (including creating an OLE object).
`pattern`	Displays the pattern palette.
`polygon`	Displays the polygon palette.
`printSetup`	Displays the Print Setup dialog box, from which the user can choose the current

	printer.
`readerRightClick`	Toggles the `sysReaderRightClick` property, which determines whether the right-click menus appear at Reader level.
`regular`	Changes a font style to `regular` (no italics or bold).
`replace`	Displays the Replace dialog box to allow the user to specify replacement text when searching for text.
`resources`	Displays the Resource Manager dialog box, from which the user can import, edit, or remove resources.
`saveAsExe`	Displays the Save As .EXE dialog box, from which the user can specify the name of the .EXE file to create.
`sendMail`	Displays the Send Note dialog box, from which the user can send an electronic mail message.
`statusBar`	Toggles the display of the status bar.
`subscript`	Formats the selected text as subscripted.
`superscript`	Formats the selected text as superscripted.
`tool`	Toggles the display of the tool palette.
`toolBar`	Toggles the display of the tool bar.

Enter event and leave event messages

Message	Description
`enterApplication`	Sent when a book opens in Multimedia ToolBook's Main window.
`enterComboBox`	Sent when a combo box receives the focus.
`enterDrop`	Sent when the cursor enters the bounds of an object during a drag-and-drop operation.
`enterDropDown`	Sent when a user clicks the combo box pushbutton to display the drop-down list.
`enterMenu`	Sent just before a menu is shown.
`enterWindow`	Sent when a viewer gets the focus.
`leaveApplication`	Sent when the book in Multimedia ToolBook's Main window is closed.
`leaveComboBox`	Sent when a combo box loses the focus.
`leaveDrop`	Sent when the cursor leaves the bounds of an object during a drag-and-drop operation.
`leaveDropDown`	Sent when a combo box's drop-down list box closes.
`leaveWindow`	Sent when the target window changes.

Mouse and keyboard event messages

Message	Description
`buttonClick`	Sent when the user presses and releases the mouse button while remaining over one object.
`keyMnemonic`	Sent when the user presses an Alt+key combination not already defined as a button or menu mnemonic access key.

Standard messages

Message	Description

`autoScript`	Displays the Auto-Script dialog box, from which the user can insert a prewritten script.
`convertPicture`	Converts the currently selected picture object to a paint object.
`customEdit`	Sent when the user clicks the Custom Edit button in an object's right-click menu. If the current system books include MTB30.SBK, this message displays the Property Browser.
`debugScript`	Displays the Debugger.
`editScript`	Displays the Script editor.
`pageSize`	Displays the Page Size dialog box, from which the user can choose a new size for the current book or background.
`showGrid`	Toggles the visibility of the grid.
`sizeToViewer`	Contracts or expands the current background to the size of the viewer in which it is displayed.
`snapGrid`	Toggles whether objects snap to the grid.

Notification messages

Message	**Description**
`linkSysBook`	Sent when a book is added to the `sysBooks` property.
`menuItemSelected`	Sent when a menu item is selected.
`pageScrolled`	Sent when the page scroll of a viewer changes to indicate how far a page has been scrolled.
`selectChange`	Sent when the user selects an item from a combo box's drop-down list box.
`selectionChanged`	Sent when the object selection changes at Author level.
`shown`	Sent when a viewer is shown.
`sized`	Sent when a viewer is sized.
`unlinkSysBook`	Sent when a book is removed from the `sysBooks` property.

Drag-and-drop messages

Message	**Description**
`allowDrag`	Sent to query whether the object can be dragged.
`allowDrop`	Sent to query whether the object will allow another object to be dropped on it.
`beginDrag`	Sent to the dragged object when a drag-and-drop operation begins.
`endDrag`	Sent to the dragged object when it is dropped.
`enterDrop`	Sent to an object when the drag cursor enters its bounds.
`leaveDrop`	Sent to an object when the drag cursor leaves its bounds.
`objectDropped`	Sent to the destination object when a dragged object is dropped on it.
`stillOverDrop`	Sent continuously to an object while the drag cursor is above it.

OLE messages

Message	**Description**
`insertOLEObject`	Displays the Insert OLE Object dialog box, which lists the types of OLE servers available.
`oleAction`	Executes the specified action for the selected OLE object.
`oleLinks`	Displays the OLE Links dialog box, which allows users to

	maintain links for OLE objects.
`pasteSpecial`	Displays the Paste Special dialog box, from which the user can create an OLE object by pasting it from the Clipboard.

Viewer messages

Message	Description
`closeWindow`	Sent to a viewer when it is closed.
`enterMenu`	Sent to a viewer when it is displayed.
`enterWindow`	Sent to a viewer when it is activated.
`hidden`	Sent to a viewer when it is hidden.
`leaveWindow`	Sent to a viewer when it loses the focus.
`menuItemSelected`	Sent to a viewer when the user chooses a menu item.
`moved`	Sent to a viewer when it is repositioned.
`newViewer`	Displays the New Viewer dialog box, from which the user can create a new viewer.
`openWindow`	Sent to a viewer when it is opened.
`shown`	Sent to a viewer when it is shown.
`sized`	Sent to a viewer when it is resized.
`viewers`	Displays the Viewers dialog box, from which users can choose a viewer to edit or create a new one.

1.41. New properties

Multimedia ToolBook 3.0 includes the following new properties. For more details about each property, see the OpenScript Reference Manual or the online Help.

Books, pages, and backgrounds

Property	Description
`buildCacheFile`	Specifies whether a CD-ROM-based application uses a cache file for speed.
`cacheFileType`	Specifies the size of the cache file for a CD-ROM-based application.
`CDMediaPath`	Specifies the directories on a CD-ROM in which Multimedia ToolBook searches for media files.
`ftsAdditionalText`	Specifies additional text for a page (beyond that in fields, record fields, buttons, and combo boxes) to be included in a full-text search index.
`ftsContext`	Specifies the context of a page for a full-text search index.
`ftsIgnore`	Specifies whether the page is to be excluded from a full-text search index.
`ftsIndexName`	Specifies the name of an index settings file used when creating a full-text search index.
`ftsKeywords`	Specifies the keywords with which users can locate the page during a full-text search.
`ftsSetFile`	Specifies the path for the index settings file used when creating a full-text search index.
`ftsTitle`	Specifies a page's title for a full-text search index.
`ftsTitleOverride`	Specifies whether the page's title for a full-text search index overrides the

	default title specified for the book as a whole.
HDMediaPath	Specifies the directories on a hard disk in which Multimedia ToolBook searches for media files.
hotwordColor	Specifies the default color for hotwords.
hotwordStyle	Specifies the default style (`frame`, `color`, or `none`) in which Multimedia ToolBook displays hotwords.
keepMenubar	Specifies whether the current menu bar should be retained when opening a new book.
notifyObjects	Lists objects on a page that have requested notification of messages reaching the page.
percentFreeSpace	Specifies the approximate percentage of free space available on a page or background.
saveOnClose	Specifies how changes are saved when the book is closed.
shownBy	Lists the viewers currently showing a specified page.
skipNavigation	Prevents Multimedia ToolBook from displaying the page at Reader level when the user navigates to it using the Page menu or using a script with implicit navigation commands (`send next`, `send previous`, and so on).
solidColorsEnabled	Specifies whether a book uses solid colors or dithered colors.
windows	Lists the viewers in a book.

Buttons

Property	**Description**
checkedGraphic	Specifies the graphic resource (bitmap, cursor, or icon) that is displayed when a checkbox-style button is checked.
disabledGraphic	Specifies the graphic resource that is displayed when a button is disabled.
ftsExclude	Specifies whether the text of the button should be excluded from a full-text search index.
ftsSection	Specifies the section in a full-text index to which the button belongs.
invertGraphic	Specifies the graphic resource that is displayed when an enabled button is clicked and held down.
normalGraphic	Specifies the graphic resource that is displayed when a button is enabled and not being held down.

Lines

Property	**Description**
lineEndSize	Specifies the size of a line's arrowhead and other ends.
lineEndStyle	Specifies the style of a line's ends (filled or open arrowheads and tails).

Fields and record fields

Property	**Description**
richText	Specifies the RTF version of text, which includes information about character and paragraph formatting.

`ftsExclude`	Specifies whether the text of the field or record field should be excluded from a full-text search index.
`ftsSection`	Specifies the section in a full-text index to which the field or record field belongs.

Hotwords

Property	**Description**
`enabled`	Specifies whether the hotword's script runs when a user clicks the hotword.
`ftsTag`	Specifies the name of the hotword's tag for a full-text search index.
`ftsTagValue`	Specifies an override value for the hotword in a full-text search index.
`hotwordStyle`	Specifies how Multimedia ToolBook should display hotwords (`frame`, `color`, `dotted`, `underlined`, or `none`).

Groups

Property	**Description**
`autoRadioButtons`	Specifies whether the radio buttons in a group operate as mutually exclusive buttons.

Viewers

Property	**Description**
`authorStatusBar`	Specifies whether a viewer will display a status bar by default at Author level.
`imageBuffers`	Specifies the number of buffers (0, 1, or 2) available to store pages and backgrounds for the current viewer.
`readerStatusBar`	Specifies whether a viewer will display a status bar by default at Author level.

Multiple objects

Property	**Description**
`defaultAllowDrag`	Specifies whether an object is dragged by default when a user clicks it.
`defaultAllowDrop`	Specifies whether an object accepts a drop by default.
`dragImage`	Specifies the graphic resource (bitmap, cursor, or icon) that is displayed when the object is dragged.
`enabled`	Specifies whether the object can respond to mouse clicks and receive the focus.
`noDropImage`	Specifies the graphic resource (bitmap, cursor, or icon) that is displayed when the object is dragged over an object that does not accept a drop.
`notifyAfterMessages`	Lists the messages for which the object has requested notification with `notifyAfter` handlers.
`notifyBeforeMessages`	Lists the messages for which the object has requested notification with `notifyBefore` handlers.
`rgbFill`	Specifies the object's fill color using RGB values (instead of HLS values).

`rgbStroke`	Specifies the object's stroke color using RGB values (instead of HLS values).

Multimedia

Property	Description
`sysMediaBreakKey`	Specifies a key that can be pressed to stop media played with the `wait` parameter.
`sysMediaSuspend`	Specifies whether Multimedia ToolBook displays an Execution Suspended error if a noncritical error occurs while a media command is playing.
`sysMMEngineVersion`	Specifies the version of the Multimedia ToolBook engine used to play media.
`sysOpenMedia`	Specifies a list of all currently open media.
`sysSupportedMedia`	Specifies the types of media devices supported on the current computer.

System

Property	Description
`caretLocation`	Specifies the location of the insertion point within a field or record field.
`startup3DInterface`	Specifies whether Multimedia ToolBook displays dialog boxes using a three-dimensional effect by default.
`sys3DInterface`	Specifies whether Multimedia ToolBook currently displays dialog boxes using a three-dimensional effect.
`sysToolBookDirectory`	Returns the path from which Multimedia ToolBook was started.

1.42. New DLL features and Windows messaging interface

* **New functions**. Extend your application with new functions in the DOS (formerly File), Windows, and Dialog DLLs.

* **Database functions**. Call the Paradox database engine to create and access Paradox database tables and indexes using the new Paradox DLL.

* **Resource handles**. Manipulate resources with Windows API functions by getting a handle for the resource with the `GDIHandle()` function.

* **Windows messages**. Extend your ability to intercept Windows messages with an improved `translateWindowMessage` control structure, which gives you complete control over how and when a message is sent to a window. You can intercept any Windows message and prevent it from reaching its window, and you can add your own custom processing for any Windows message. To make it easier for you to find the right Windows message, Multimedia ToolBook includes an online Help application called WINCONST.HLP that lists Windows constants.

1.43. New DLL functions

Multimedia ToolBook 3.0 includes the following new DLL functions. For more details about each function, see the OpenScript Reference Manual or the online Help.

TB30DLG.DLL

```
addComboBoxItem()
addListBoxItem()
chooseColorDlg()
chooseDirectoryDlg()
chooseFontDlg()
colorPaletteDlg()
controlIDToName()
controlNameTohWnd()
controlNameToID()
deleteComboBoxItem()
deleteListBoxItem()
deletenComboBoxItem()
deletenListBoxItem()
dialogCallback()
enableControl()
endTBKDialog()
enterWaitState()
getControlText()
getCustomColors()
getDialogFocus()
getListBoxItems()
getListBoxSelection()
getnListBoxSelection()
getOpenFileDlgFilterIndex()
getSaveAsDlgFilterIndex()
isButtonChecked()
isControlEnabled()
leaveWaitState()
openFileDlg()
saveAsDlg()
setBitmapData()
setButtonCheck()
setComboBoxItems()
setControlText()
setCustomColors()
setDialogFocus()
setGroupedButtonCheck()
setIconData()
setListBoxItems()
setListBoxSelection()
setnListBoxSelection()
```

TB30DOS.DLL

```
getCDDriveList()
getDirectoryOnlyList()
getDriveKind()
getFileOnlyList()
getVolumeName()
isCDDRive()
setFileDate()
setSystemDate()
setSystemTime()
```

TB30WIN.DLL

```
getIniVar()
getModuleList()
```

```
getModulePath()
popText()
popTextGetBounds()
sendKeys()
setIniVar()
```

MTB30MM.DLL

```
HMSfromMillisec()
millisecFromHMS()
millisecFromMSF()
millisecFromSMPTE()
MSFfromMillisec()
SMPTEfromMillisec()
tbkMMTranslate()
```

2.0. Converting books from earlier versions

2.1. How to convert books from earlier versions

The Multimedia ToolBook 3.0 file format has changed from Multimedia ToolBook 1.5. To use your existing 1.5x books in Multimedia ToolBook 3.0, you must convert them.

Important Once a book is converted to the new file format, you will not be able to open or otherwise access it in Multimedia ToolBook 1.5.

To convert your 1.5x books:

1. Back up all your books before opening them in Multimedia ToolBook 3.0.

   Multimedia ToolBook automatically backs up a version of each book it converts and gives it the extension .OLD. However, for extra safety you should save your own backup copy of each file.

   Be especially careful to back up all system books and other books that are accessed from within your application.

2. Confirm that you have enough free space on your hard disk to save a duplicate copy of your books.

3. Open each book individually.

   When you open your Multimedia ToolBook 1.5x books in Multimedia ToolBook 3.0, your books are automatically converted to Multimedia ToolBook's new file format. This process may take some time if your book is large.

If your application contains multimedia elements, see "Converting multimedia applications" for details about converting to Multimedia ToolBook 3.0.

Recompiling scripts

Multimedia ToolBook 3.0 features an updated compiler. To bring scripts up to date, Multimedia ToolBook recompiles them automatically.

If errors are reported during the conversion, check the conversion log file to determine the source of the error. We recommend that you then run the Script Walker utility (SCRWALK.TBK) and open each book to update all your scripts in one pass. Remember to update your system books as well. (You do not need to update system books shipped with Multimedia ToolBook 3.0, such as MTB30.SBK).

Note If your Multimedia ToolBook 1.5 scripts use syntax no longer supported by Multimedia ToolBook 3.0, they cannot be recompiled until you fix the syntax errors.

Keyword conflicts

While converting books from Multimedia ToolBook 1.5x, Multimedia ToolBook 3.0 automatically adds an at symbol (@) to the beginning of any user-defined property, function, or variable in your scripts that conflicts with a new keyword. This ensures that your scripts will continue to execute correctly.

Note If scripts cannot be converted (for example, if they use syntax no longer allowed in Multimedia ToolBook), Multimedia ToolBook marks the script with a special code to indicate that it is an old script that was not converted. Edit the script to remove syntax errors. Multimedia ToolBook will then automatically detect that it has been marked and add the @ operator to words that conflict with new keywords.

Multimedia ToolBook cannot automatically add an at symbol to the following keywords because they are used with DLL declarations:

BYTE
DWORD
DOUBLE
FLOAT
INT
LONG
POINTER
STRING

If these words appear in your script as the names of variables or user properties, you must manually add the at symbol as a prefix or change the name.

Books with stripped scripts

Multimedia ToolBook cannot convert books whose script text has been removed with the Script Remover (REMOVER.EXE) utility provided in the Multimedia ToolBook 1.5 Developer Utilities. Multimedia ToolBook recompiles all scripts when it converts books to the new file format, so the original script text must be present. If you open a book whose scripts have been removed, Multimedia ToolBook converts the objects.

2.2. Maintaining old and new versions of Multimedia ToolBook

When you install Multimedia ToolBook 3.0, you can continue working with earlier versions of Multimedia ToolBook. Keep these points in mind when using more than one version of Multimedia ToolBook:

* If you convert a book to Multimedia ToolBook 3.0 format, you can no longer use it with earlier versions of Multimedia ToolBook. If you want to use a book with both Multimedia ToolBook 1.5 and Multimedia ToolBook 3.0, you must make a copy for each version.

* Multimedia ToolBook executables (.EXE files) and DLLs have new names in Multimedia ToolBook 3.0 so that they don't conflict with older versions of these files. You can continue to use older file names in the scripts of your Multimedia ToolBook 1.5 applications; when you convert applications to Multimedia ToolBook 3.0, update them by substituting new file names.

* You can launch the appropriate version of Multimedia ToolBook automatically when you click on a .TBK file in the Windows File Manager or use the Program Manager Run command to run a .TBK file. If you associate the new utility TBLOAD.EXE with your .TBK files, it can detect which version of Multimedia ToolBook created the file and launch it. For details, see Associating TBK files with Multimedia ToolBook 1.5 and 3.0.

2.3. Converting multimedia applications

You can convert the multimedia aspects of applications created in Multimedia ToolBook 1.5 to Multimedia ToolBook 3.0 in one of three ways:

* **Continue to call the multimedia DLL functions**. Instead of using the system book TBKMM.SBK, use the system book MTB30MM.SBK. This system book links the multimedia functions you used in your application (the functions are now in MTB30MM.DLL instead of TBKMM.DLL). Do not use the version of TBKMM.SBK provided with Multimedia ToolBook 1.5.

* **Use the `callMCI()` function**. Substitute calls to the `callMCI()` function for the `tbkMCI()` calls in your application. Because `callMCI()` is a native OpenScript function, you can execute MCI commands without requiring a system book to link DLL functions.

* **Rewrite using new multimedia features**. Rewrite the multimedia portions of your application to take advantage of the new capabilities in Multimedia ToolBook 3.0, including clips, stages, and OpenScript commands such as `mmPlay`. This is the recommended method; although the conversion effort is more involved, you can add new features to your application, and it becomes more flexible and robust.

2.4. Recompiling scripts created with Early Access versions of Multimedia ToolBook

Multimedia ToolBook 3.0 features an updated compiler. To bring scripts up to date, Multimedia ToolBook recompiles them automatically if it detects that the script was compiled using the older compiler. Books created with Early Access will initially experience a slowdown as Multimedia ToolBook encounters and recompiles scripts not yet updated. We recommend that you run the Script Walker utility (SCRWALK.TBK) and open each book to update all your scripts in one pass. Remember to update your system books as well (you do not need to update system books shipped with Multimedia ToolBook 3.0 such as MTB30.SBK).

3.0 Features that work differently in Multimedia ToolBook 3.0

3.1. Changed keyboard accelerators

A number of Multimedia ToolBook's keyboard accelerators have changed. To view the new keyboard accelerator assignments, browse through the menus.

You can now move the edit cursor one word at a time in a field by pressing the Ctrl+Left Arrow or Ctrl+Right arrow keys. To accommodate this change, Multimedia ToolBook's new keyboard accelerators for page navigation are as follows:

Navigation	Old accelerator	New accelerator
Previous Page	Ctrl+Left	Alt+Left
Next Page	Ctrl+Right	Alt+Right
First Page	Ctrl+Up	Alt+Up
Last Page	Ctrl+Down	Alt+Down

Note To move from word to word in the Command window, use Alt+arrow keys.

3.2. Evaluating expressions in the Command window

Because of the new assignment operator (=), you can no longer evaluate expressions by typing them in the Command window by themselves. Instead, enter a full command such as the following:

```
put sqrt(540)
request uniqueName of this book
```

3.3. Starting Runtime Multimedia ToolBook without a file name

You can start the Runtime version of Multimedia ToolBook 3.0 without specifying a file name in the command line. In Multimedia ToolBook 1.5, executing TBOOK.EXE without a file name would result in an error message and Multimedia ToolBook would not start. If you start Runtime Multimedia ToolBook 3.0 (MTB30RUN.EXE) without a file name, Multimedia ToolBook displays an Open dialog box from which you can choose a book.

3.4. ID numbers of objects during recording

As in Multimedia ToolBook 1.5, Multimedia ToolBook 3.0 inserts the keyword `selection` into a script as you are recording it. However, if you use the right-click menu to change an object's properties while recording, Multimedia ToolBook inserts the object's actual ID number. You should examine scripts you record with the script recorder in Multimedia ToolBook 3.0 to be sure that this change does not affect how the recorded script runs.

3.5. Change in file format

Multimedia ToolBook 3.0 features a new file format for books (.TBK and .SBK files). Files from older versions of Multimedia ToolBook are converted automatically when you open them using Multimedia ToolBook 3.0, or when you refer to an object in them.

Important After you convert a book using Multimedia ToolBook 3.0, you cannot open it using an earlier version. Be sure to make backup copies of books that you intend to use with both versions.

3.6. Using the MTB30.INI and ASYM.INI files

Multimedia ToolBook 3.0 keeps startup information in the MTB30.INI file (in contrast to Multimedia ToolBook 1.5, which kept this information in a [TOOLBOOK] section of the WIN.INI file).

Information about graphic filters is now stored in the ASYM.INI file, which can be shared with other Asymetrix applications such as MediaBlitz! and Compel. Multimedia ToolBook 1.5 stored this information in the [ToolBook Filters] section of the WIN.INI file.

3.7. New executable and DLL file names

The names of the Multimedia ToolBook executable (.EXE) files and DLLs have been changed in Multimedia ToolBook 3.0, so you can continue using your Multimedia ToolBook 1.5 applications without changing names or scripts that link DLLs. When you convert your application to Multimedia ToolBook 3.0, however, you should update scripts that link DLLs to use the new DLL names.
The old and new names appear in the tables below. For information about what files to including when you are shipping your application, see "Shipping files with Runtime Multimedia ToolBook."

Development version files

Old name	New name
tbkbase.dll	mtb30bas.dll
tbkbmpt.dll	mtb30bmp.dll
tbkcomp.dll	mtb30cmp.dll
tbkedit.dll	mtb30edt.dll
tbkmm.dll	mtb30mm.dll
tbkmm.ini	mtb30mm.ini
tbknet.exe	mtb30net.exe
tbknet.exe	mtb30net.exe
tbkutil.dll	mtb30utl.dll
toolbook.exe	mtb30.exe

Optional runtime files

Old name	New name
tbkdb3.dll	tb30db3.dll
tbkdlg.dll	tb30dlg.dll
tbkwin.dll	tb30win.dll
tbkfile.dll	tb30dos.dll

Runtime Multimedia ToolBook files

Old name	New name
tbkrun.exe	mtb30run.exe
tbkbase.dll	mtb30bas.dll
tbkutil.dll	mtb30utl.dll
tbkcomp.dll	mtb30cmp.dll
tbkmm.dll	mtb30mm.dll

3.8. MTB30NET.EXE required for all file access

Multimedia ToolBook now requires the MTB30NET.EXE file for all file access. In previous versions of ToolBook, this file was required only if you were coordinating access to a book by multiple users on a network. Be sure to include the MTB30NET.EXE file any time you distribute an application.

3.9. Global variable functions

The Multimedia ToolBook 1.5 functions listed below are still included in MTB30BMP.DLL for backward compatibility. However, they are linked automatically only if you make MTB30MM.SBK a system book. Otherwise, you must link the functions manually.

```
    clearAllGlobalVars()
    getGlobalVar()
    globalVarCount()
    setGlobalVar()
```

3.10. Importing picture graphics

When you imported any graphic that created a picture in Multimedia ToolBook 1.5, the graphic was imported at a fixed size, regardless of its natural size. In Multimedia ToolBook 3.0, graphic files are imported at their true size, which can be as large or larger than the Main window. If you want to import graphic files and create fixed-size picture objects out of them, use a script such as the following:

```
importGraphic "C:\GRAPHICS\SAMPLE.TIF"
get size of selection
size of selection = 1920,1920*(item 2 of It/item 1 of It)
```

3.11. `visible` property of groups

In Multimedia ToolBook 1.5, if you set a group's `visible` property to `true`, the `visible` property of all the objects in the group was set to `true` as well, and they became visible. If you set the group's `visible` property to `false`, all objects in the group disappeared as well.

Multimedia ToolBook no longer passes the group's `visible` property to individual objects in the group. If you hide a group, all objects in it disappear. However, if you show a group, only the objects whose `visible` property is `true` appear. If an object's `visible` property is `false`, it remains hidden.

This change may affect Multimedia ToolBook 1.5 applications that show or hide all objects in a group by showing or hiding the group. To work around this problem, you can show or hide the objects of the group. For example, these Multimedia ToolBook 1.5 statements show or hide all objects in a group:

```
show group id 23
hide group id 23
```

To do the same in Multimedia ToolBook 3.0, use statements such as these:

```
show objects of group id 23
hide objects of group id 23
```

If you have nested groups, you must execute these commands on all the nested groups in turn.

3.12. Changes in wordwrap behavior

Multimedia ToolBook 3.0 wraps text slightly differently than Multimedia ToolBook 1.5. All text now wraps the same way on the printed page as it does on screen.

If you notice that text in fields or button captions is wrapping and clipping differently, resize your fields and buttons to regain the same wrapping and clipping of text as in Multimedia ToolBook 1.5.

3.13. New Author menu

The Multimedia ToolBook 3.0 Author-level menu bar is changed from Multimedia ToolBook 1.5. Scripts that modify the Author-level menu bar may not work correctly if they refer to menus or items that no longer exist.

In addition, menus are now resources that can differ from book to book. For backward compatibility, Multimedia ToolBook 3.0 provides the `keepMenuBar` property that specifies whether the menu bar should be retained when you switch between books.

3.14. Author menu item in Runtime Multimedia ToolBook

Runtime Multimedia ToolBook no longer automatically removes the Author menu item from the default menu bar resource assigned to the Main window. To remove it, use the OpenScript `remove menuItem` command.

You may want to remove the Author menu item from all menus, even in the full version of Multimedia ToolBook. Because the F3 accelerator key is always available, you do not require the Author menu item in order to switch to Author level in the full version of Multimedia ToolBook.

3.15. Extended user property and variable references

Multimedia ToolBook 3.0 allows you to add the at symbol (@) prefix to the names of user properties, user-defined functions (`to get` or `to set` handlers), or variables to distinguish them from OpenScript keywords.

The following statements illustrate using @ to distinguish a user property name from a keyword:

```
text of field id 0 = "this goes into the text of the field"
@text of field id 0 = "this goes into a user property"
get @text of field id 0
```

The following example illustrates using @ for variable references. Without the prefix, Multimedia ToolBook would generate a syntax error because `style` is a property name:

```
local @style
@style = 1
```

There are over 180 new keywords that may conflict with user properties or variable names you used in Multimedia ToolBook 1.5. For example, one of Multimedia ToolBook's new keywords is `enabled`; if you had a user property named `enabled` in Multimedia ToolBook 1.5, you must now use the @ prefix to get or set its value:

```
get @enabled of button "Quit"
set @enabled of button "Quit" to false
```

3.16. Passing `null` to a DLL

Multimedia ToolBook 3.0 treats `""` and `null` differently when you use them as string parameters to a DLL function. If you specify `""`, Multimedia ToolBook passes a pointer to an empty string ("`\0`"). If you specify `null`, it passes a null pointer (`0:0`). In contrast, Multimedia ToolBook 1.5 passes a pointer to an empty string in both cases.

This change makes it easier to pass a null pointer to DLLs that distinguish between a pointer to an empty string and a null pointer (for example, the Windows `FindWindow()` function).

To ensure complete compatibility with Multimedia ToolBook 1.5, change your scripts to use `""` instead of `null` when passing empty strings to DLL functions. If you do not make this change, you may encounter a general protection fault when a script passes `null` to a DLL.

If you write your own DLLs, be sure a string parameter is a null pointer before dereferencing it. Otherwise, your DLL may cause a general protection fault when Multimedia ToolBook passes `null` to it.

3.17. Stricter OpenScript syntax

Multimedia ToolBook 3.0 enforces OpenScript syntax more rigorously than Multimedia ToolBook 1.5 did. When you save your scripts, the OpenScript compiler does not allow some syntax that was allowed in Multimedia ToolBook 1.5. You may not be able to save some of your scripts without making minor modifications to eliminate syntax errors.

To check the syntax of an individual script, display the script in the Script editor, then choose Check Syntax from the File menu.

To review all scripts in the book in one pass, run the Script Walker utility included with Multimedia ToolBook 3.0 by double-clicking its icon from the Multimedia ToolBook 3.0 Program Manager group. Script Walker examines all of the scripts in a converted book and finds those that contain syntax errors.

3.18. Changes in `to get` handlers

Unlike Multimedia ToolBook 1.5x, Multimedia ToolBook 3.0 requires every code path in a `to get` handler to contain a return statement. If a `to get` handler does not contain the necessary `return` statements, its script will not compile when the book is converted. You can fix this by editing the script and adding `return` statements at appropriate locations.

For example, you might be missing a `return` statement in a `conditions` control structure that doesn't have a default branch and has no `return` statement after it. Even if you provide a `return` statement for each branch, Multimedia ToolBook requires that you provide a default `return` statement of some kind. The following handler is missing a `return` statement:

```
to get calcVal x
  conditions
     when x is 1
        return 2
     when x is 2
        return 3
  end conditions
end calcVal
```

The following two scripts illustrate corrected versions of this handler.

```
to get calcVal x
  conditions
     when x is 1
        return 2
     when x is 2
        return 3
     else
        return 0 --return added to default branch
  end conditions
end calcVal

to get calcVal x
  conditions
     when x is 1
        return 2
     when x is 2
        return 3
  end conditions
  return 0 -- return added after conditions
end calcVal
```

You might also be missing `return` statements in `if/then` control structures. If the only `return` statement is inside an `if/then` control structure that doesn't have an `else` branch, you must add at least one additional `return` statement following the `if/then` control structure.

3.19. Redeclaring variables in handlers

Multimedia ToolBook 3.0 does not allow you to declare the same variable twice in the same handler. You should delete duplicate variable declarations. If you're using the `local` statement to clear a local variable, use the `clear` command instead.

3.20. `Null` versus `none`

Multimedia ToolBook 3.0 does not allow you to use `null` and `none` interchangeably, as you could in some circumstances in Multimedia ToolBook 1.5. For example, these two statements are equivalent in Multimedia ToolBook 1.5:

```
set borderStyle to null
set borderStyle to none
```

The actual value is `none`, but Multimedia ToolBook 1.5 automatically converts `null` to `none`. In Multimedia ToolBook 3.0 you must replace such uses of `null` with `none`.

3.21. Changed error messages

A number of error messages in Multimedia ToolBook 3.0 have changed, so an error in Multimedia ToolBook 3.0 may return a different error message than the same error in Multimedia ToolBook 1.5. If your application checks for particular error messages, test carefully to ensure that your error checking still operates correctly.

3.22. Obsolete OpenScript properties, messages, commands, and functions

The following OpenScript terms are obsolete in Multimedia ToolBook 3.0. Some of the terms still work for backward compatibility with Multimedia ToolBook 1.5, but you should update your scripts to use the new terms. For more information about the new terms, see the OpenScript Reference Manual and online Help.

Properties

Term	Replaced with	Still works?
`caption` book property	`caption of mainWindow`	yes
`captionShown` book property	set caption of `mainWindow` to `space`	no
`icon` book property	`icon of mainWindow`	yes
`sysMagnification`	`magnification` viewer property	yes
`sysMousePosition`	`mousePosition` viewer property	yes
`sysRuler`	`rulers` viewer property	yes

Messages

Term	Replaced with	Still works?
`align` menu event message	`align<Type>` messages	no
`palettes` menu event message	(use new messages that toggle each individual palette)	no
`printerSetup` menu event message	`printSetup`	no
`toggleStatus` keyboard event message	`statusBar` message	no
`windowMoved` notification message	`moved`	no
`windowShown` notification message	`shown`	no
`windowSized` notification message	`sized`	no

Commands and functions

Term	Replaced with	Still works?
`activate menuItem` command	`enable menuItem`	yes
`deactivate menuItem` command	`disable menuItem`	yes
`menuState()` function	`menuEnabled, menuItemChecked, menuItemEnabled`	yes
`before|after` with `translateWindowMessage`	(forward from called handler)	yes

Multimedia DLL functions

The following multimedia-related OpenScript terms were DLL functions in Multimedia ToolBook 1.5, but are now native OpenScript functions. You can continue to use the older functions by using the `MTB30MM.SBK` system book with your application.

Function	Change
`tbkMCI()`	`callMCI()`
`tbkBitmap()`	`imageCommand()`
`tbkTimerCapability()`	`timerCapability()`
`tbkTimerStart()`	`timerStart()`
`tbkTimerStop()`	`timerStop()`
`tbkMMSignal`	`MCISignal`
`tbkMMNotify`	`MCINotify`
`tbkMMTimer`	`timerNotify`

3.23. Changed OpenScript commands, functions, and messages

Commands

Command	Change
`local <array>`	Declares the dimensions of an array.
`system <array>`	Declares the dimensions of an array that is used as a system variable.

Functions

Function	Change
`round()`	Accepts a parameter to specify the number of decimal places at which to round.
`objectFromPoint()`	Accepts a viewer reference.

Messages

Message	Change
`moved`	Sends a message when a viewer is moved.

4.0. Documentation additions

The following topics describe information that became available after we printed the Multimedia ToolBook documentation.

4.1. Associating .TBK files with Multimedia ToolBook 1.5 and 3.0

To manage .TBK files created with different versions of Multimedia ToolBook (1.5, and 3.0), use the File Manager to associate .TBK files with the TBLOAD.EXE utility. When you double-click any Multimedia ToolBook file or use the RUN command in File Manager, TBLOAD.EXE decides which version of Multimedia ToolBook was used to create the book and launches the appropriate version. To establish the proper paths for different versions of Multimedia ToolBook, add a section to your ASYM.INI file based on this example:

```
[ToolBook Load Information]
TB15=C:\TOOLBOOK\TOOLBOOK.EXE
TB30=C:\TB30\TB30.EXE
MTB30=C:\MTB30\MTB30.EXE
```

4.2. Creating and accessing databases using Multimedia ToolBook

Multimedia ToolBook allows you to create and maintain databases using two sets of DLL function calls: one to maintain a dBASE III-compatible database, and another to maintain a Paradox database. These database DLLs enable you to combine Multimedia ToolBook's graphical user interface with sophisticated database management.

To learn about using the database DLLs with Multimedia ToolBook, refer to the Database Reference online Help system provided separately with Multimedia ToolBook.

4.3. Distributing Multimedia ToolBook applications

A new tool, Asymetrix Setup Utility, has two parts: SETUPMGR.TBK and SETUP.EXE. SETUPMGR.TBK allows you to package your application into a set of compressed files you can ship to customers. Your users can then use SETUP.EXE to decompress and install the files on disk.

Asymetrix Setup includes features to

* package multiple books, DLLs, and additional files into a single application.
* create optional modules in your application.
* create Program Manager groups and items.
* assign Program Manager icons to individual files.
* launch additional applications after installation is complete.

For details about Asymetrix Setup Utility, launch SETUPMGR.TBK from the Multimedia ToolBook 3.0 Program Manager group, then choose Help from the menu.

4.4. Sending electronic mail from Multimedia ToolBook

You can use your electronic mail (e-mail) system directly from Multimedia ToolBook by choosing Send Mail from the File menu at Author level. For this menu item to function, you must be using an e-mail system compatible with the Windows Mail API (MAPI), such as Microsoft Mail.

To use e-mail from OpenScript, send the `sendMail` message.

You can also use OpenScript programs to invoke e-mail from Multimedia ToolBook. For an example of how to do this, see the MAPI.TBK sample application.

4.5. Printing color pages on black-and-white printers

Multimedia ToolBook 3.0 provides enhanced color conversion for printing to monochrome and grayscale printers. The effect of this enhancement is that color pages printed to black-and-white printers from Multimedia ToolBook 3.0 will look significantly better than they do in Multimedia ToolBook 1.5.

When printing colors to a black-and-white printer, Multimedia ToolBook scales the colors to gray using a value called the "gamma factor," which specifies how much Multimedia ToolBook lightens each color. The value of the gamma factor can be between 1 and 255; the lower the number, the more colors are lightened. The default is 125.

To change the gamma factor, specify a new value for the "GammaFactor" entry in the MTB30.INI file. For example, to lighten colors further, change the entry in the MTB30.INI file to a value such as this:

```
GammaFactor=100
```

4.6. Asymetrix System Information utility

You can use a utility called Asymetrix System Information to display the following information about your computer:

* Processor type
* Memory available
* Disk drives available and the amount of free space on each one
* Windows directory
* Color device information (such as the number of colors and the driver name)
* Drivers installed for multimedia

Run Asymetrix System Information as a standalone utility from the Program Manager or by choosing Run from the File menu in Multimedia ToolBook 3.0. You can also add it as a menu item on the Help menu in Multimedia ToolBook by adding MTB30.SBK to the `sysBooks` property.

4.7. Using multiple instances of Multimedia ToolBook

With Multimedia ToolBook 3.0's ability to create viewers (multiple windows), most applications can run successfully using only a single Multimedia ToolBook instance (copy). However, in some circumstances you might want to start a second instance of Multimedia ToolBook as part of your application. If you do, here are a few tips:

* You do not need to open another instance of Multimedia ToolBook in order to get and set properties of objects in other books. You can simply refer to the property of an object from another book, and Multimedia ToolBook will access the book and get the value without starting another instance.

* Persistent properties apply across instances. If you load the same book in two or more instances of Multimedia ToolBook, any persistent property of an object is reflected in all running instances of that book. For example, if a user is typing in a field, each change to the field's text shows up in all instances.

* Settings for nonpersistent properties of all objects apply only to the current instance. For example, if you set `sysGrid` to `true` in one instance, it is not necessarily true in the other instance. Viewers have a special set of nonpersistent properties that allow them to be used effectively in multiple instances.

* You can run one instance of Multimedia ToolBook at Author level while another is at Reader level. This allows you to send commands to the Reader-level instance if you cannot get to Author level in that instance.

See also
ToolBook User Manual, Chapter 11, "Creating windows with viewers"

4.8. Using Multimedia ToolBook on a network

Multiple users can share Multimedia ToolBook applications on a network if the Multimedia ToolBook utility MTB30NET.EXE is in each user's path. This utility comes with Multimedia ToolBook and is installed in the same directory as Multimedia ToolBook itself.

MTB30NET guards against two or more users changing a book at the same time. If that were to happen, either one user's changes would be lost or the book could become corrupted.

There are two ways to maintain network versions of Multimedia ToolBook applications:

* Provide a read-only version of a book that can be opened and used by anyone. However, because the book is read-only, no changes can be saved, including text changed in fields.

* Provide a standard, read-write version of a book, which can be opened by only one user at a time. No other user can open the book (even implicitly by attempting to get the property of an object in the book). Other users cannot copy the book, delete it, or change its file attributes until the first user closes it.

To make a book read-only, use the Windows File Manager. Select the file, choose Properties from the File menu, then check the Read Only box under Attributes.

Tip For best performance, each user should run Multimedia ToolBook locally and use the network only to access the book itself.

4.9. Limits of Multimedia ToolBook objects

The following table lists the limits to which various Multimedia ToolBook objects are constrained.

Book
Maximum number of pages: 65,536 (64K). This maximum is reduced when the book contains viewers, scripts with more than 1,024 bytes, bitmap resources, menu bars, color palettes, or embedded OLE objects. In practical terms, a book is limited to about 50,000 pages.

Background, page

Maximum number of objects on background or page: varies. Each background or page can occupy up to 65,536 bytes (64K) of memory. Each object on the background or page, including pages, requires a fixed amount of memory. (For example, each button requires about 90 bytes.) Additional memory is occupied by text (in fields, record fields, and combo boxes), scripts, and user properties.

The maximum number of objects therefore depends on which objects are placed on the background and how much extra memory they require. As an example, you could put 1,100 lines on a background if none of them had scripts or user properties. However, you could only place two fields on the background if each contained 32,000 characters of text.

To optimize the way a background or page uses memory, save the book with a different name with the `save as` command, which organizes the memory used by a background or page to its optimal configuration.

Field, record field

Maximum number of characters: 32,000. If the text is formatted, or if there are other objects on the background or page, you might not be able to put the maximum amount of text into the field or record field.

Variable

Maximum number of characters: 65,473. Maximum data in all variables combined: 16MB.

Handlers

Maximum levels of nesting: 1.024 (1K), but this limit is reduced depending on the complexity of the script. The practical limit is between 50 and 300.

Tip To determine how much free space is available on a background or page, use `percentFreeSpace` property.

4.10. Features not supported in Runtime Multimedia ToolBook

The following tables list the properties, commands, and messages that do not function in Runtime Multimedia ToolBook. Unless otherwise noted, attempting to use these results in an error.

At the end of this topic you will find a list of additional differences in behavior between the development and Runtime versions of Multimedia ToolBook.

Properties

Attempting to set any of the following properties in Runtime Multimedia ToolBook results in an error. You can get the value of some of the properties, as listed in the table.

Property	**Value in Runtime Multimedia ToolBook**
`bounds of commandWindow`, `toolBar`, any palette	
`position of commandWindow`, `toolBar`, any palette	
`size of commandWindow`, `toolBar`, any palette	
`startupAutoScriptFile`	Name of Auto-Script file
`startupReaderRightClick`	`false`
`style of toolBar`, `toolPalette`	
`sysAutoScriptFile`	Name of Auto-Script file
`sysGrid`	`false`
`sysGridSnap`	`false`

`sysGridSpacing`		
`sysLevel`		`reader`
`sysReaderRightClick`	`false`	
`sysRuler`		`false`
`sysRuntime`		`true`
`sysShowMRUFiles`		`false`
`sysTool`		`reader`
`tile` of `toolbar`, `toolPalette`		
`tileOrder of toolbar`, `toolPalette`		
`tileWrap of toolbar`, `toolPalette`		
`visible of commandWindow`	`false`	
`visible of toolbar`, any palette	`false`	

Commands
Attempting to execute the following commands results in an error in Runtime Multimedia ToolBook:

`edit script`
`sort` (but not the `sort` message)
`put` command with `commandWindow` as the destination

Runtime Multimedia ToolBook does not respond when you execute the following commands, but it does not display an error message:

```
add menu
add menuItem
check menu
check menuItem
disable menu
disable menuItem
enable menu
enable menuItem
remove menu
remove menuItem
restore menubar
uncheck menu
uncheck menuItem
```
Any menu command that specifies `at author`

Messages
Runtime Multimedia ToolBook displays an error if you attempt to send any of the following messages that display an Author-level dialog box, manipulate the palettes or the grid, or change to Author level.

```
author
autoScript
backgroundProperties
bookProperties
color
command
contents
debugScript
editScript
```
`file1, file2, file3, file4` (MRU file list aliases)
```
grid
importGraphic
keyboardShortCuts
```

```
learningToolBook
line
lineEnds
menuCommands
newViewers
openScriptReference
pageProperties
pageSize
pattern
polygon
properties
reader (ignored; no error)
resources
rulers
searchForHelpOn
showGrid
snapGrid
sort
startRecording
stepByStep
stopRecording
technicalSupport
tool
toolbar
viewers
```

Additional differences in Runtime Multimedia ToolBook

* Help buttons are removed in all built-in dialog boxes.

* Instead of displaying the Execution Suspended message box when encountering an error, Runtime Multimedia ToolBook displays the text of the error message in an Error message box. The user can see more information about the error by holding down the Ctrl key while clicking Cancel. Runtime Multimedia ToolBook displays the Execution Suspended dialog box with all buttons removed except Cancel.

* If you start Runtime Multimedia ToolBook without a file name

  - Multimedia ToolBook displays the Open dialog box to prompt you for the name of a .TBK file to open.

  - Runtime Multimedia ToolBook displays the copyright bitmap (no copyright bitmap is displayed if you specify a book on the command line).

  - the default menu bar is the File menu with Open and Exit menu items.

4.11. Available multimedia drivers

The Multimedia ToolBook 3.0 CD-ROM contains drivers for the following multimedia types:

* Video for Windows version 1.1 (.AVI files)
* AutoDesk Animator version 1.1 (.FLI and .FLI files)
* QuickTime for Windows version 1.1.1 (.MOV, .PIC, and .JPG files)
* Pioneer laserdisc players

* VISCA-compatibile VCR devices

These files are located in the \drivers directory on the CD-ROM. MCI drivers for additional multimedia devices are available from the device manufacturer.

With the exception of QuickTime for Windows, these drivers may be redistributed with your Multimedia ToolBook application. For information regarding licensing the QuickTime drivers for redistribution, please see the README.WRI file in the QTW directory.

Installing the multimedia drivers

Video for Windows version 1.1 (.AVI files)

* Run SETUP.EXE in the \DRIVERS\VFW11 directory on the Multimedia ToolBook CD-ROM.

AutoDesk Animator version 1.1 (.FLI and .FLI files)

* Run SETUP.EXE in the \DRIVERS\AUTODESK directory on the Multimedia ToolBook CD-ROM.

QuickTime for Windows version 1.1.1 (.MOV, .PIC, and .JPG files)

* Run SETUP.EXE in the \DRIVERS\QTW directory on the Multimedia ToolBook CD-ROM.

Pioneer laserdisc players

1. In the Windows Control Panel, click the Drivers icon to display the Drivers dialog box.
2. Click Add.
3. Under List of Drivers in the Add dialog box, choose Unlisted or Updated Driver.
4. In the Install Driver dialog box, click Browse, then select the \DRIVERS\LASERDISK directory on the Multimedia ToolBook CD-ROM.
5. Click OK twice.

To configure the laserdisc driver for your system, click Setup in the Drivers dialog box.

VISCA-compatible VCR devices

1. In the Windows Control Panel, click the Drivers icon to display the Drivers dialog box.
2. Click Add.
3. Under List of Drivers in the Add dialog box, choose Unlisted or Updated Driver.
4. In the Install Driver dialog box, click Browse, then select the \DRIVERS\VISCA directory on the Multimedia ToolBook CD-ROM.
5. Click OK twice.

4.12. Shipping files with Runtime Multimedia ToolBook

The following files are required for Runtime Multimedia ToolBook:

MTB30BAS.DLL
MTB30BMP.DLL
MTB30CMP.DLL
MTB30FLT.DLL
MTB30LNL.DLL
MTB30MM.DLL
MTB30MM.INI
MTB30NET EXE

MTB30RUN.EXE
MTB30UTL.DLL
PCDLIB.DLL
PCDXBMP.DLL
PHOTO.DLL
TBLOAD.EXE

The following files are optional; you can include them with your application if necessary:

PXENGWIN.DLL	(used with Paradox Engine DLL)
TB30DB3.DLL	(used with dBASE III DLL)
TB30DLG.DLL	
TB30DOS.DLL	
TB30PDX.DLL	(used with the Paradox Engine DLL)
TB30WIN.DLL	

Note For information about the Paradox Engine and dBASE III DLLs, see "Creating and accessing databases using Multimedia ToolBook."

If you are shipping an application containing a full-text search index, you should include the following files as well:

FTS30IQR.DLL
FTS30RDR.DLL
FTS30HLP.STS
FTS30MAC.STS
FTS30MSG.STS
FTS30MTB.DLL

For information about what files you are allowed to distribute with your Runtime Multimedia ToolBook application, refer to the file called FILELIST.WRI.

4.13. Searching for Multimedia ToolBook from .EXE files

When you create an .EXE file using the Save as EXE command, the book contains instructions on launching Multimedia ToolBook (MTB30.EXE). The book searches for MTB30.EXE in this order:

(1) The directory specified in the TB30 entry of the [Registered Apps] section in the ASYM.INI file.

The book searches first for the development version of Multimedia ToolBook (MTB30.EXE), then the Runtime version (MTB30RUN.EXE).

(2) The DOS search paths.

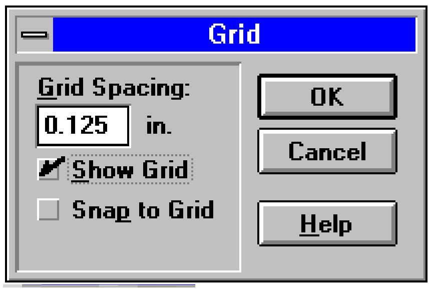
(3) The program associated with .TBK files in the [Extensions] section of the WIN.INI file. If you perform a full installation of Multimedia ToolBook, the associated application will be TBLOAD.EXE.

4.14. Specifying three-dimensional controls in dialog boxes

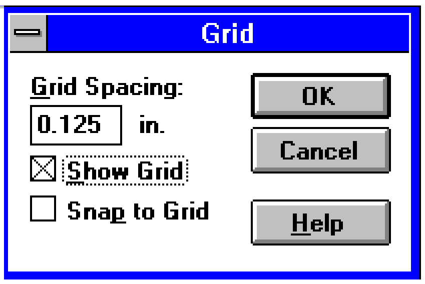
The built-in dialog boxes that appear in Multimedia ToolBook can appear in the traditional Windows style or the newer three-dimensional style. You specify the style of dialog boxes using the system property `sys3DInterface`. If this property is `true`, all dialog boxes appear in the three-dimensional style; if `false`, all dialog boxes appear in the traditional style.

You can set Multimedia ToolBook's default value for this property by setting the corresponding `startup3DInterface` to `true` or `false`.

Example of a three-dimensional dialog box



The same dialog box in the traditional style



4.15. Global accelerator keys

To switch from Author to Reader level and back again, you can choose Reader or Author from the Edit menu, or you can press the F3 accelerator key. The F3 accelerator key is always available at Reader level. Therefore, you can remove the Author menu item from the menu of the Main window or of your viewer and still be able to switch to Author level.

When you press F3, Multimedia ToolBook sends the `author` message to the current page in the window that has the focus. If there is no handler in the object hierarchy to interpret the message, Multimedia ToolBook switches to Author level.

Note You cannot switch from Reader to Author level in Runtime Multimedia ToolBook, so the F3 accelerator key is not available in that version.

Disabling the F3 key

To disable the F3 key, write a handler for the `author` message that does not forward the message. For example:

```
to handle author
  request "You cannot switch to Author level in this book."
end
```

Note Use this approach with care. If you trap the `author` message, you will not be able to switch to Author level at all, even if you retain the Author menu item.

Defining additional accelerator keys

To define other accelerator keys, specify them as part of the menu bar for a viewer (window). For details about creating or editing a menu bar, start the Menu Bar editor by choosing Menu Bar under Available Resources in the Resource Manager dialog box, then clicking New. Press F1 in the Menu Bar editor for help on how to define accelerator keys.

Alternatively, you can write a `keyUp` handler to respond to key event message for particular keys. For example, this handler in a book script allows you to edit the script of any object when you press Ctrl+E:

```
to handle keyUp key, isShift, isCtrl
  if key = KeyE and isCtrl = true then
     Ask  "What object?"
     edit script of It
  else
     forward
  end
end
```

4.16. Menu event messages sent from tool bar

When you click a button on the tool bar, Multimedia ToolBook sends the corresponding menu event message to the page in the viewer that has the focus. For example, if you click the New Page button, Multimedia ToolBook sends the `NewPage` message.

Tip To use the tool bar in a viewer other than the Main window, tear it off the Main window by clicking between buttons and dragging it away from the frame. It will then operate with whatever viewer has the focus.

See also
ToolBook User Manual, Chapter 2, "ToolBook orientation"
ToolBook User Manual, Chapter 21, "Messages, objects, and properties"

4.17. Tool palette and tool bar captions and wrapping behavior

You can right-click the tool palette and the tool bar to display a menu that allows you to specify whether you want to see graphics, captions, or both. Both can also be wrapped when they're tiled. The wrapping is optional and can be set by right-clicking the tool bar or tool palette, or by setting a new property called `tileWrap` to true.

See also
ToolBook User Manual, Chapter 2, "ToolBook orientation"

4.18. Adding keywords to a page using OpenScript

You can add keywords to a page using the Page Settings box. Alternatively, you can use an OpenScript command such as the following:

```
push "Pigeons" onto ftsKeywords of this page
```

There is no equivalent command to create a list of keywords for the book as a whole. If you add keywords to pages using this command, they do not initially appear in the Keywords in Book(s) list box when you display the Page Settings dialog box or in the Keywords Used in Index list in the Keywords tab of the Index Settings dialog box.

However, you can build the list out of existing keywords. In the Index Settings dialog box, click the Files tab, then remove and re-enter the name of the current book from the list under Included Books. When you click OK, Multimedia ToolBook rebuilds the list of keywords currently available in the book.

4.19. Adding pages to contexts using OpenScript

You can add a page to a context using an OpenScript command such as the following:

```
ftsContext of this page = "owls"
```

There is no equivalent command to create a list of contexts for the book as a whole. If you add a page to a context using this command, the context does not initially appear in the Page Contexts list in the Pages tab of the Index Settings dialog box.

However, you can build the list out of existing contexts. In the Index Settings dialog box, click the Files tab, then remove and re-enter the name of the current book from the list under Included Books. When you click OK, Multimedia ToolBook rebuilds the list of contexts currently available in the book.

4.20. Objects in unnamed sections

If you do not explicitly assign an object on a page to a section, Multimedia ToolBook assigns it to a default section called OTHER. You can search this section as you would any other.

To change the name of the default section, edit the index settings file (.SST file) that defines the index. In the [CREATE] section, change the following line, substituting a new name for the word `OTHER`:

```
DEFAULT_SECTION=OTHER,YES
```

4.21. Overriding word separators

By default, when Multimedia ToolBook builds an index for full-text search, it considers alphabetic characters (A-Z, a-z) part of a word, and numeric characters (0-9) if they appear in a word next to alphabetic characters. Any other characters, including spaces and punctuation, are considered white space that separates words. For example, if your application contains file names with paths (such as \APPS\MTB30), Multimedia ToolBook separates each component into separate words ("APPS" and "MTB30")

You can override Multimedia ToolBook's definition of a word separator. For example, if you want "\APPS\MTB30" to be indexed as a single word, you can specify that the backslash is part of the word.

To treat characters as part of a word instead of as separators, edit the index settings file (.SST file) that defines the index. In the [CREATE] section, add the characters to the SPECIALS setting. For example, to treat a backslash as a word characters, use this command:

```
SPECIALS=\
```

If you want to add more than character, list them one after another with no spaces or other characters in between, as in this example that includes an underscore:

```
SPECIALS=/_
```

Avoid treating the following characters as word characters, because they are reserved in search expressions:

* comma (,)
* plus sign (+)
* minus sign/hyphen (-)
* equal sign (=)
* at-sign (@)

Note Be sure that you want every instance of the character treated as part of a word before assigning it to the SPECIALS item.

4.22. Overriding hotword tag values

By default, the value of a Hotword tag is the text of the hotword itself. However, you can override the value by assigning an alternative value for that tag. This is useful to help you index consistent values for a Hotword tag, such as using only a singular form of a word or using a consistent date format.

To override the value of a hotword tag, assign its `ftsTagValue` property:

```
ftsTagValue of hotword "01 Jan 94" = "01:01:94"
```

4.23. 32-instance limit on viewers

There is a 32-instance limit on opening and closing viewers in one script (a single handler and any other handlers that it calls). This limit applies to opening and closing the same viewer repeatedly or opening and closing up to 32 separate viewers.

When Multimedia ToolBook is idle, it clears the count of viewer instances.

See also
ToolBook User Manual, Chapter 11, "Creating windows with viewers"

4.24. Creating hotwords from RTF text

You can create hotwords using a word processing or other Windows program that supports rich-text format (RTF), then pasting or importing them into your application:

To create a hotword from RTF text:

1. Create the text in a word-processing program that can cut or copy RTF text onto the Clipboard.

2. Format the text that will be a hotword with a double underline.

3. Format at least one character immediately following the double-underlined text as hidden text

4. Close the hotword definition with a semicolon (;).

5. Cut or copy the text to the Clipboard.

6. In Multimedia ToolBook, put the insertion point into a field and paste the text. Text that is double-underlined with a following hidden character becomes a hotword.

Tip Instead of cutting the text to the Clipboard and pasting it in Multimedia ToolBook, you can use the Import Text option from the Tools menu.

You can specify the `name` and `hotwordstyle` properties for the hotword by including parameters for them in the hidden text following the double-underlined text. Use this format, where the single underline represents hidden text:

```
NewHotword name=Contents; hotwordstyle=frame;
==========-------------------------------------
```

The space immediately following `newHotword` is hidden as well. You can include additional user properties and values (up to 100 characters), but any properties other than `name` and `hotwordstyle` will be treated as user properties.

See also
ToolBook User Manual, Chapter 6, "Adding text"

4.25. Editing resources using resource editors

Multimedia ToolBook includes a number of editors you can use to create new resources (bitmaps, color palettes, menu bars, icons, and cursors) or edit existing ones. To create a new resource, click New in the Resource Manager dialog box. To edit an existing resource, select the resource in the Resource Manager dialog box and click Edit. Multimedia ToolBook starts the appropriate resource editor:

When editing	Multimedia ToolBook starts
Bitmaps	BitEdit (BITED30.EXE)
Color palettes	PalEdit (PALED30.EXE)
Menu bars	MenuEdit (MENUED30.EXE)
Icons	Icon/Cursor editor (ICONED30.EXE)
Cursors	Icon/Cursor editor (ICONED30.EXE)

You can also run these editors as standalone applications.
For help while using the resource editors, press F1.

4.26. Changes to the File menu

If you invoke a resource editor from the Resource Manager dialog box, Multimedia ToolBook changes the resource editor's File menu to reflect the commands that are available while working within Multimedia ToolBook. The File menu is laid out as shown below (<type> menu item refers to the type of resource supported by the editor: Bitmap, Palette, Menubar, Icon, or Cursor).

```
File
   New
   Open From...
       <Type> In Current Book...
       <Type> In Other Book...
       File...
   Save | Update
   Save As...
       <Type> In Current Book...
       <Type> In Other Book...
       File...
   --------------------
   Exit
```

New
Creates a new resource after prompting you to save or update the current resource. If a current book is defined, it is closed. In the Icon/Cursor editor, the New Resource dialog box allows you to specify what type of resource to create.

Open From <Type> In Current Book
Opens a resource for editing from those already imported into the book you are working with. This menu item is only enabled if you invoked the editor from a Multimedia ToolBook book.

Open From <Type> In Other Book
Opens a resource for editing from a book other than one you are working with. The book you choose becomes the current book and the Choose Resource dialog box is displayed, from which you can choose the resource to edit.

Open From File
Open a resource from a DOS file. For example, you can open and edit a color palette as a DOS file that has a .PAL extension. If you are working in a book, it is closed.

Save | Update Menu Item
Saves or updates the current resource. If you have not previously saved the resource, or if you are editing a resource as a DOS file, this menu item is Save. If you are editing a resource already imported into a book, the menu item is Update.

Selecting this menu item for a new, unsaved resource is the same as selecting Save As File.

Save As <type> In Current Book
Saves the resource you are working with as a new resource in the current book. When you choose this option, you are prompted to assign a name to the resource.

Save As R<type> In Other Book

Saves the resource as a new resource in a book other than the one you are working with. When you choose this option, you are prompted to assign a name to the resource. The book you are working with is closed, and the new book becomes the current book.

Save As File
Saves the resource as DOS file. If you are working with a book, the book is closed.

Exit
Close the application. If you have not saved your work, you are prompted to do so.

See also
ToolBook User Manual, Chapter 14, "Using resources"

4.27. Creating clips from Kodak Photo CD images

You can create a clip to display photographs from a Kodak Photo CD. This is different from importing a Photo CD image, which converts the image into a bitmap. Creating a clip of a Photo CD image enables you to position and size the photograph using a stage, display it with OpenScript commands, and change its attributes by setting clip properties.

To create a clip from a Kodak Photo CD:

1. From the Object menu, choose Clips to open the Clip Manager.
2. Click New to open the Choose Source Type dialog box, then select Image (Photo CD).
   The Photo CD dialog box appears.
3. Under CD Drive, select the letter of the drive that contains the Photo CD.
   A thumbnail of the first photo on the CD appears
4. To move between photos, use the scroll bar at the bottom of the picture. To speed the process of scrolling, click Load All to preload the thumbnails into memory.
5. When you have selected the photo, choose its size from the Size list box. (You can resize the image in a stage when you display it.)
6. Click OK to create the clip.
7. In the Clip editor Clip box, type a name for the image.
8. Click OK to close the Clip editor, then click Close to close the Clip Manager.

Tip If you intend to distribute the Photo CD with your application, be sure you have appropriate rights. For information about the rights for any particular image, click Rights in the Photo CD dialog box.

4.28. Importing graphics

In Multimedia ToolBook 3.0 you can import graphics from a variety of formats. The ASYM.INI file lists the filters that are available.

File	**Source file format**	**ToolBook object type**
.BMP	Bitmap	bitmap
.DIB	Device Independent Bitmap	bitmap
.CDR	CorelDRAW	picture
.CGM	Computer Graphics Metafile	picture
.DRW	Micrografx Draw Picture	picture
.DXF	Autodesk AutoCAD Interchange	picture
.EPS	Encapsulated PostScript	picture
.GIF	Graphics Interchange Format	picture
.CH2	Harvard Graphics 2.x	picture
.CH3	Harvard Graphics 3.0	picture
.PCT	Macintosh QuickDraw Picture	picture
.PCX	ZSoft PC Paintbrush	picture
.PIC	Lotus 1-2-3 Graphic	picture
.SY3	Harvard Graphics 3.0 Symbol	picture
.SYM	Harvard Graphics 2.3 Symbol	picture
.TIF	Tagged Image File Format	picture
.WMF	Windows Metafile	picture

Multimedia ToolBook creates a picture out of any graphic file format that requires an import filter. However, you can convert picture objects to paint objects using the new `convertPicture` message.

Quality of imported graphics

You might detect a difference in how well Multimedia ToolBook displays a graphic depending on how you have imported it. For example, if you import a Windows metafile (.WMF file) by copying it to the Clipboard and then pasting it onto a Multimedia ToolBook page, it might not look as good as if you had imported it using the Import Graphic command on the File menu. If the quality of an imported graphic is not what you expected, try one of these solutions:

* Export the graphic from the original source as a .WMF file, then use the Import Graphic command on the File menu to import it.

* Import the graphic from the source application into some other application (such as a word processing program); if it looks good there, copy it to the Clipboard, then paste it in Multimedia ToolBook.

If the colors in the graphic are not what you expect, you may be importing a graphic into Multimedia ToolBook that was created with more colors than your computer can display. For example, if the graphic was created with 32,000 colors but your system only uses 256, Multimedia ToolBook dithers the 32,000 colors using the current color palette, which may result in colors that don't match the original graphic very well. You can adjust the colors by using a different color palette in the book, or by modifying the graphic in the source application to match the number of colors available on your system.

See also
ToolBook User Manual, Chapter 10, "Adding graphics and images"
ToolBook User Manual, Appendix B, "Working with 256-color display devices"

4.29. OLE server busy message

If the OLE server is busy when you try to edit or play an OLE object, Multimedia ToolBook enters a wait state in which you cannot click the mouse or enter keystrokes. If the server is still unavailable after six seconds, Multimedia ToolBook displays a dialog box that lets you switch to the application that is the OLE server or retry your operation. (By switching to the application, you may be able to correct a condition, such as an open dialog box, that is keeping it from becoming available as an OLE server.) If the server becomes free, the dialog box disappears and your operation continues.

See also
ToolBook User Manual, Chapter 12, "Using object linking and embedding"

4.30. Coordinating linked OLE objects with closed servers

If your Multimedia ToolBook book contains a linked OLE object and the OLE server closes its view of the linked files, you may experience two problems.

1. If the OLE server closes its view of the linked file without saving changes, your OLE object might still reflect these discarded changes.

   To keep your OLE object current, set its `upToDate` property to `true` at strategic times, such as when you navigate to the page that displays the object, when you leave that page, or when you close the book. Keep in mind, though, that setting an OLE object's `upToDate` property to `true` invokes the OLE server if it is not already running.

2. If the OLE object is in a book that doesn't currently appear in the Main window, the OLE object may keep that book open until it is notified that the OLE server has closed its view of the linked file.

   Each time you set the `upToDate` property of an OLE object, it confirms its connection to the OLE server. You can therefore temporarily set the OLE object's `upToDate` property to `false`; if the OLE object determines that the OLE server has closed its view of the linked object, the OLE object allows the Multimedia ToolBook book to close.

See also
ToolBook User Manual, Chapter 12, "Using object linking and embedding"

4.31. Displaying modal viewers

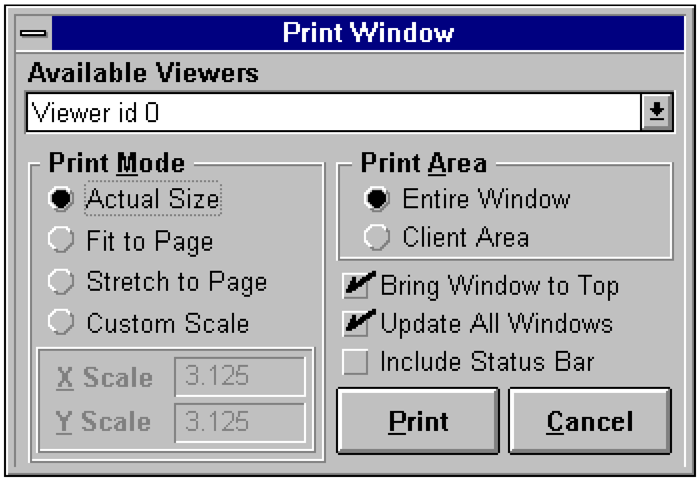
You cannot display modal viewers at Author level. If you display a modal viewer at Reader level and then switch to Author level, the viewer becomes modeless. If you are at Author level when you try to display a modal viewer, Multimedia ToolBook displays an error.

See also
ToolBook User Manual, Chapter 11, "Creating windows with viewers"

4.32. Printing viewers

A new system book, PRINTWND.SBK, allows you to print a viewer and its current contents. Adding PRINTWND.SBK to the `sysBooks` property adds a Print Window item to the File menu at the Author level (which sends the `printWindow` message).

When you choose this menu item, Multimedia ToolBook displays this dialog box:


Print Window

Available Viewers: Viewer id 0

Print Mode: [✓ Actual Size] Fit to Page Stretch to Page Custom Scale

X Scale 3.125

Y Scale 3.125

Print Area: [✓ Entire Window] Client Area

☑ Bring Window to Top

☑ Update All Windows

☐ Include Status Bar

Available Viewers
Choose the viewer to print from the list of all viewers currently open. The viewer you choose determines what area of the screen is printed.

Print Mode
Specifies how the viewer is scaled to the printed page.

Actual Size. Prints the viewer at the size it appears on the screen.
Fit to Page. Scales the viewer up to the best fit on the page.
Stretch to Page. Stretches the viewer to fill the entire page.
Custom Scale. Scales the viewer by the factors you specify in the X Scale and Y Scale boxes.

Print Area
If Entire Window is checked, Multimedia ToolBook prints the contents of the viewer as well as its frame and caption bar. If Client Area is checked, Multimedia ToolBook prints only the area inside the viewer's frame.

Bring Window to Top
If this option is checked, Multimedia ToolBook places the viewer to be printed on top of other windows before printing. Multimedia ToolBook also hides the tool bar, palettes, and the Command window before printing.

Note Multimedia ToolBook cannot place a viewer on top of its own child windows. For example, if the viewer you want to print displays a child window as a tiled tool palette, both viewers print, even if you check the Bring Window to Top box.

Update All Windows
Specifies that all windows currently visible should be refreshed before Multimedia ToolBook prints the viewer. This option ensures that if Multimedia ToolBook hides the Command window and tool palettes, any objects behind them print correctly.

Include Status Bar
If checked, specifies that the viewer's status bar (if any) is printed.

X and Y Scale boxes

If you choose Custom Scale under Print Mode, enter the factors (X for horizontal dimension and Y for vertical dimension) by which you want the viewer scaled. For example, an X factor of 3.125 will cause the viewer to be printed three and one-eighth times wider than normal.

4.33. Clearing and resetting arrays

The `clear` command deletes both fixed and dynamic arrays. After an array is cleared, it is no longer accessible in the current handler. The `clear` command is useful for freeing the memory used by an array. You should clear large arrays, especially system arrays, when they are no longer used.

If you want to preserve the dimensions of a dynamic array but clear all of its entries, use the `fill` command to set all elements of the array to `null`. For example:

```
fill myArray with null
```

An undocumented command, `reset`, resets the dimensions of a dynamic array to zero, which is useful if you want to replace contents of the array with new contents containing fewer elements. Resetting the array saves memory and allows the `dimensions()` function to return an accurate count of the number of elements. For example:

```
local largeArray[], smallArray[]
largeArray[20] = "sample text"
smallArray[1] = "more text"
request dimensions( largeArray )     --Returns 20
reset largeArray             --Removes all elements
largeArray = smallArray
request dimensions( largeArray )      --Returns 1
```

Multimedia ToolBook displays an error if you attempt to reset a fixed array.

See also
ToolBook User Manual, Chapter 23, "Variables, values, and text"

4.34. Declaring arrays as fixed or dynamic

You can declare an array as `fixed` or `dynamic`. This eliminates the ambiguity that arises when you use arrays for system variables and arguments, and it provides more rigorous error checking when arrays are passed between handlers. For example:

```
to handle buttonClick
  local array1[10]
  local array2[]
  get userFunction(array1, array2)
end

to get userFunction fixed arrayVar[], dynamic arrayVar2[]
  --Further statements here
end
```

The `fixed` keyword in this example specifies that `arrayVar` is a fixed array of one dimension with an unspecified number of elements. Because the array is fixed, you cannot perform any dynamic array operation on it, such as resetting or expanding it. If you declare an argument to be a fixed array, but you pass it as a dynamic array, the argument is still treated as a fixed array, and you cannot expand or reset it. The keyword `dynamic` specifies that the array `arrayVar2` is a dynamic array and can be reset.

System variables may also be declared as `fixed` or `dynamic`. You can declare a fixed system variable whose size you do not know in advance. For example:

```
system fixed bar[]
system fixed string alphaArray[]
system dynamic int numArray[]
system int numArray2[]      --dynamic is the default
```

Multimedia ToolBook will check your array system variables to ensure they are compatible with any previously declared version of the same variable. For example, if you try to redeclare a 10-element fixed array as a 5-element fixed array, or redeclare a dynamic array as a static array of different dimensions, Multimedia ToolBook displays an error.

See also
ToolBook User Manual, Chapter 23, "Variables, values, and text"

4.35. Linking or embedding OLE objects using OpenScript

You can paste an OLE object in OpenScript by copying the object from the OLE server to the Clipboard, then sending the `pasteSpecial` message with one of these Clipboard format parameters:

* `linked` to create a linked object
* `embedded native` to create an embedded object

For example:

```
--Creates a linked object from the object on the Clipboard
send pasteSpecial linked
```

See also
ToolBook User Manual, Chapter 12, "Using object linking and embedding"

4.36. Optimizing comparisons to `true` and `false`

If you are determining whether a value is `true` or `false`, you can optimize the comparison by avoiding a string comparison with the string `true`. For example:

```
--As documented
if visible of selection = true then ...
--Faster version of same comparison
if visible of selection then ...
```

See also
ToolBook User Manual, Chapter 22, "Statements, functions, and expressions"

4.37. Passing arrays by reference

By default, when you pass an array as a parameter to another handler, the array is passed by value, meaning that Multimedia ToolBook makes a copy of it for the new handler. Changes made to the array in the called handler are not reflected in the calling handler, which retains its own copy of the array.

To provide optimal speed and programming convenience, you can also pass an array by reference: the calling and called handlers share the same array. Changes made to the array in the called handler are reflected in the calling handler automatically. Passing arrays by reference is preferred for tasks such as sorting arrays.

To pass an array by reference, add the phrase `by reference` in the parameter list of the called handler. For example:

```
to get swapElements myArray[] by reference, elem1, elem2
   temp = myArray[elem1]
   myArray[elem1] = myArray[elem2]
   myArray[elem2] = temp
   return true
end
```

Because you are passing the array `myArray` by reference, the swap operation performed in this handler swaps the original elements in the calling handler.

See also
ToolBook User Manual, Chapter 23, "Variables, values, and text"

4.38. Unlinking DLL functions

You can now unlink individual DLL functions instead of unlinking the entire DLL. Use either of these formats:

```
unlinkDLL function <function list> from <dll name>
unlinkDLL functions <function list> from <dll name>
```

The `<function list>` parameter contains an expression that evaluates to a list of one or more function names. The `<dll name>` parameter contains the name of the DLL from which to unlink the functions. For example:

```
unlinkdll function "messageBeep" from "user"
unlinkdll functions "messageBeep,GlobalAlloc" from "kernel"
```

Determining what DLL functions are already linked
You can determine which DLL functions are linked by calling the `DLLFunctions()` function. Use the following syntax:

```
DLLfunctions(<dll name>)
```

The `<dll name>` parameter contains the name of the DLL for which you want to determine what functions are linked. The function returns a comma-separated list of the functions linked in the specified DLL. If the DLL is not linked, the function returns `null`. For example:

```
get DLLfunctions("tb30dlg.dll")
```

4.39. Undocumented OpenScript keywords

The following OpenScript keywords have been added since the OpenScript Reference Manual was finished.

Commands and functions

Keyword	Description
`by reference`	Qualifier for passing arrays as parameters to other handlers.
`fixed, dynamic`	Qualifiers for declaring arrays with the `local` and `system` commands.
`getFileListDlg()`	DLL function that displays the standard Windows File List dialog box.
`getFileListDlgIndex()`	DLL function that returns the index number of a file selected in the File List dialog box.
`getFileVersion()`	DLL function that returns information about a file, including its current path, version number, copyright, and product name.
`isMenu(), isMenuItem()`	Determines whether a menu or menu item exists.
`listToTextline()`	DLL function that converts comma-delimited lists to CRLF-delimited textlines.
`sortList()`	DLL function that sorts lists of items.
`sortText()`	DLL function that sorts CRLF-delimited lines.
`textlineToList()`	DLL function that converts CRLF-delimited textlines to comma-delimited lists.

Messages

Keyword	Description
`stateChanged`	Sent to a viewer when it is minimized, restored, or maximized.
`systemRestored`	Sent when the Restore System command is executed. Useful to alert system books that they must reinitialize system variables and relink DLLs

Properties

Keyword	Description
`onSave`	Additional value for `updateLink` property of linked OLE objects that specifies that the object is to be updated when the user saves the book containing it.
`tileWrap`	Tool bar and tool palette property. If `true`, and if the object (tool bar or tool palette) is tiled, the object wraps when the window is made narrower than the object. If `false`, the object is clipped.
`wordWrapUnits`	Book property that specifies horizontal and vertical spacing for text.

5.0. Documentation corrections

The documentation set contains a small number of technical errors. Typographic errors that do not affect the accuracy of information are not noted in this list. For details about a particular error, click the Correction text.

5.1. ToolBook User Manual

Page	Correction
GS-5,6	Using the Multimedia ToolBook tutorial
GS-10	What's New file
1-6+	Sample application pictures
3-26	Adding a draw object to a group
4-7	Importing books
4-16	Creating passwords
5-5	Excluding buttons from the tabbing order
10-6	Bounds of paintObject
11-12	SkipNavigation property
11-30	menuBar property
11-39	Clarification of send and to get with the in control structure
11-47	Displaying a dialog box using viewers (example)
12-13	Setting OLE updateType
12-19	Editing an embedded object
12-23	PrimaryAction and secondaryAction keywords
13-6	Menus as resources
13-25	Disable menuItem and enable menuItem
13-33	menuBar parameter in popMenu( )
14-4	Extracting color palettes from resources
14-15	Exporting resources
15-8	printerStyle value
15-13	Print Header and Print Footer dialog boxes
15-15,-16	Data format of text
15-17	Mismatched field names
16-5	Importing fixed-field files
16-12	Importing pages
17-7	Setting up Runtime Multimedia ToolBook
21-9	Using parameters with messages
22-11	Parameters in handlers
22-18	Graphic operator
23-6	Maximum size of long integer
23-7	Color data type
23-14	Passing arrays by value
24-28	verticalDisplayRes( ) and horizontalDisplayRes( )
A-11	Finding strings with the is in operator

5.2. OpenScript Reference Manual

Page	Correction
1-4+	Size property
1-54	ReaderStatusBar viewer property
2-23	AutoSize viewer property
2-47	clear command
2-62	CreateHotword message
2-70	DefaultPosition of viewers
2-122	GDIHandle() function
2-145, -149	International system properties
2-216	ParentHandle property
2-263	RichText property
2-280	seekFile command

2-315	StoredImages property
2-347	TargetWindow property
2-356	Style of target
2-362	Transition command
3-19	GetListBoxSelection() function
3-55, 3-56	GetModuleList(), getModulePath()
A-14	TextFromPoint() function

5.3. Multimedia ToolBook Manual

Page	Correction
6-17	sysError at idle
6-31	CDMediaPath in comment
10-9	HDMediaPath property
10-17	mmBackgroundPalette property
10-34	mmStatus property
10-18	mmClose command
10-24	mmPause command
10-28	mmRewind command
10-36	mmStop command

Using the Multimedia ToolBook tutorial

Where: ToolBook User Manual, pages GS-5, GS-6

Another way to start the tutorial is to double-click the Multimedia ToolBook Tutorial icon in the Multimedia ToolBook Program Manager group.

Step 2 suggests that you click an option in the Contents screen to learn about a particular feature. Instead, you should click the Map button in the Introduction screen for Unit 1 to see a list of units and the lessons in them.

What's New file

Where: ToolBook User Manual, page GS-10

There is no file called What's New; information about new features and converting applications appears in this Help application.

Sample application pictures

Where: ToolBook User Manual, Chapter 1

The pictures in Chapter 1 of the tutorial application do not reflect the actual Multimedia ToolBook 3.0 tutorial.

Adding a draw object to a group

Where: ToolBook User Manual, page 3-26

Step 1 of the instructions for adding a draw object should read:

1. Double-click any object in the group to which you want to add a new object.

Importing books

Where: ToolBook User Manual, page 4-7

In the first OpenScript example, the word "book" should not be inside quotation marks. The example should read:

```
import book "c:\toolbook\add.tbk"
```

Creating passwords

Where: ToolBook User Manual, page 4-16

Multimedia ToolBook accepts passwords in any language, not just English.

Excluding buttons from the tabbing order

Where: ToolBook User Manual, page 5-5

The button option Exclude From Tab Order listed in step 3 on this page is described incompletely. When you choose this option, Multimedia ToolBook does skip the button when you attempt to tab to it or if you press its mnemonic access key.

However, Multimedia ToolBook does not ignore the tab. Instead, Multimedia ToolBook moves the focus to the next object in the layer order that can accept it. For example, if a button that is excluded from the tab order is next to a field, the focus moves to the field when you try to tab to the button. If you are not aware of this behavior, or if you have set the Exclude From Tab Order option inadvertently, the results of tabbing to the button can be confusing.

Bounds of `paintObject`

Where: ToolBook User Manual, page 10-6

In the OpenScript example, the words `paint object` should be one word. The correct example is:

```
bounds of paintObject "flower" = 100,100,2000,2000
```

`skipNavigation` property

Where: ToolBook User Manual, page 11-12

The `skipNavigation` property is ignored when you use `go` to navigate. Therefore, remove the parenthetical remark, "or the `go` command with these values" in the first paragraph of this page.

`menuBar` property

Where: ToolBook User Manual, page 11-30

The first example at the top of the page should reference the property `menuBar`, not `menu Bar`. The correct example is:

```
menuBar of viewer "Map" = menu ID 1
```

Clarification of `send` and `to get` with the `in` control structure

Where: ToolBook User Manual, page 11-39

The `in` control structure (to change the target window) changes the context for commands but it does not change the effect of the `send` command, the object hierarchy of the target object, or the `to get` control structure.

In the `send` and `to get` commands, the target is the object itself. If the object does not contain a handler for the message, the message is forwarded up the hierarchy of that object, not up the hierarchy of the target window.

For example, imagine that you have two windows open: the Main window and a small viewer called "Quick Jump," which contains buttons called "Contents" and "Index." These buttons send the messages `contentsPage` and `indexPage`, respectively. The handlers for these messages are in the background of the page displayed in the Main window.

You might write a handler such as this for the Contents button, intending to call the handler in the Main window's background:

```
to handle buttonClick
   in viewer id 0
      send contentsPage
   end
end buttonClick
```

This won't work. Even though you explicitly set the target window to be the Main window before sending the message, the message is sent first to the Contents button, then to its page, and so on up the button's hierarchy. Because the page in the Quick Jump viewer does not share a background with the Main window, the handler for `contentsPage` is not found.

To fix the problem, send the message to an explicit target, namely the current page in the Main window. Within the `in` control structure, Multimedia ToolBook interprets implicit references to refer to the current window, so this small change will achieve the intended effect:

```
to handle buttonClick
   in viewer id 0
      send contentsPage to this page
   end
end buttonClick
```

Displaying a dialog box using viewers (example)

Where: ToolBook User Manual, page 11-47

In step 7, the first line of the `buttonClick` handler reads:

```
showPage = "dialog page"
```

It should read:

```
showPage = page "dialog page"
```

Setting OLE `updateType`

Where: ToolBook User Manual, page 12-13

In both OpenScript examples, the object reference is missing. Change the commands to something like these examples:

```
updateType of OLE ID 1 = automatic

upToDate of OLE "chart" = true
```

Editing an embedded object

Where: ToolBook User Manual, page 12-19

The Object menu item referred to in step 1 under "Editing an embedded object" does not exist. Instead, if an OLE object is selected, you see a menu item with the form "<action> <OLE server> Object," where

* <action> is the type of action associated with the object, such as "Edit."
* <OLE server> is the name of the object's OLE server.

For example, if the currently selected OLE object is a bitmap, the menu item is "Edit Paintbrush Picture Object."

If the object has more than one possible action, the menu item is "<OLE server> Object," and it displays a cascading menu of possible actions. For example, if the OLE object is a sound file, the menu item is "Sound Object", and it displays a cascading menu with the items "Play" and "Edit."

`primaryAction` and `secondaryAction` keywords

Where: ToolBook User Manual, page 12-23

The last paragraph on this page refers to entries for `primaryAction` and `secondaryAction` in Chapter 2 of the OpenScript Reference Manual. However, there are no separate entries for these keywords in the book. Instead, they are documented under `action` on page 2-5 of the OpenScript Reference Manual.

Menus as resources

Where: ToolBook User Manual, 13-6

A note at the bottom of the chart says all menu bars except the Author menu bar are editable resources. However, the Print Preview menu bar also is not editable.

`menuBar` parameter in `popMenu()`

Where: ToolBook User Manual, page 13-33

In the second line of the example, the word "menu" should be "menuBar". The line should read:

```
get popupMenu(pLoc, menuBar "rightClick", object of target)
```

`disable menuItem`, `enable menuItem`

Where: ToolBook User Manual, page 13-25

The examples for `disable menuItem` and `enable menuItem` should say "in menu," not "of menu."

Extracting color palettes from resources

Where: ToolBook User Manual, page 14-4

The second footnote below the table on this page states that Multimedia ToolBook can extract color palettes from bitmaps and Windows metafiles. However, Multimedia ToolBook can extract color palettes only from palette (.PAL) files.

Exporting resources

Where: ToolBook User Manual, page 14-15

In the OpenScript example for exporting a resource, the target file name should be in quotation marks. The correct example is:

```
export resource cursor id 100 as "point.cur"
```

`printerStyle` value

Where: ToolBook User Manual, page 15-8

In the second line of the first example, the correct printer style is "groups", not "group". The line should read:

```
printerStyle = groups
```

Print Header and Print Footer dialog boxes

Where: ToolBook User Manual, page 15-13

The dialog box for defining headers and footers has changed. To finish defining a header or footer, click Close, not OK. To cancel a header or footer, delete the text in the dialog box or set the header or footer properties to null using OpenScript.

Data format of text

Where: ToolBook User Manual, pages 15-15, 15-16

The format expression for a number of examples on these two pages is missing. On page 15-15, the example should read:

```
text of recordField "Total" > 100 as text
```

In the second line of the first example, the correct printer style is "groups", not "group". The line should read:

```
"text of recordfield "total" > 100 as text"
```

On the example next to "FIeld contains a range of numbers" on page 15-16, the second line should read:

```
"and text of recordfield "Contribution" >= 100 as text"
```

On the same page, the example next to "Field contains today's date" should read:

```
"text of recordfield "PurchaseDate" > sysdate as text"
```

Mismatched field names

Where: ToolBook User Manual, page 15-17

In the example there is a discrepancy between the field name `Hired` in the second line and the name `Hire Date` in the `printerConditions` line; these should be the same. For example, change the second line to this:

```
printerFields = "First name,Last name,Title,Hire Date"
```

Importing fixed-field files

Where: ToolBook User Manual, page 16-5

In the OpenScript example for importing fixed-field text files, the list of field lengths should be in quotation marks. The correct example is:

```
import "c:\data\customer.txt" as fixed using "3,5,8,12"
```

Importing pages

Where: ToolBook User Manual, page 16-12

In the OpenScript example for importing pages, the source file should include the extension ".TBK." The correct example is:

```
go last page
import pages 2 to 16 of book "c:\mybooks\glossary.tbk" \
    without background
```

Setting up Runtime Multimedia ToolBook

Where: ToolBook User Manual, page 17-7

The procedure described for setting up a Runtime version of Multimedia ToolBook has been superseded by the new Asymetrix Setup Utility, SETUPMGR.TBK. For details about setting up applications using this new utility, launch SETUPMGR.TBK from the Multimedia ToolBook 3.0 Program Manager group, then choose Help from the menu.

See also
Distributing Multimedia ToolBook applications
Features not supported in Runtime
New executable and DLL file names
New tools for distributing applications
Shipping files with Runtime Multimedia ToolBook

Using parameters with messages

Where: ToolBook User Manual, page 21-9

The first example program under "Using parameters with messages" uses the name "position" as a variable. Because `position` is an OpenScript keyword, the script will not compile. Substitute another variable name such as "pos" instead:

```
to handle buttonclick pos
   request "you clicked at " && pos
end
```

Tip You can also add an at symbol (@) in front of the word "position" to distinguish it from the OpenScript keyword. For details, refer to the ToolBook User Manual, Chapter 23, "Variables, values, and text."

Parameters in handlers

Where: ToolBook User Manual, page 22-11

The parameters in the first example on this page should be separated by a comma. The correct example is:

```
to get GrossMargin Sales, COGS
```

`graphic` operator

Where: ToolBook User Manual, page 22-18

The table of operators for string evaluation does not include the operator `graphic`, which assigns or returns graphics that are part of a text string. For details about this operator, see page 2-126 in the OpenScript Reference Manual or the online Help.

Maximum size of `long` integer

Where: ToolBook User Manual, page 23-6

The correct range for a variable with the data type of `long` is -2147483648 to 2147483647.

`color` data type

Where: ToolBook User Manual, page 23-7

The `color` data type will accept only HLS values, not RGB values.

Passing arrays by value

Where: ToolBook User Manual, page 23-14

In the first handler of the example script, you cannot use the `get` command with an expression that returns an array because the variable `It` is not an array. Change the fifth line of the script to read:

```
vAlphabet = invertArray( vAlphabet )
```

If you want to preserve the value of `vAlphabet`, substitute the following handler for the entire first handler in the example:

```
to handle buttonClick
  local vAlphabet[26]
  local vInvertedAlphabet[26]
  -- (fill vAlphabet[] here)
  vInvertedAlphabet = invertArray( vAlphabet )
  --(further processing here)
end
```

In addition, in the `to get` handler, you must declare the passed array as a fixed array with the keyword `fixed`. Change the first line of the `to get` handler as follows:

```
to get invertArray fixed vPassedArray[]
```

Note You can also now pass arrays by reference, which makes it easier to share arrays between handlers.

`verticalDisplayRes()` and `horizontalDisplayRes()`

Where: ToolBook User Manual, page 24-28

The functions `verticalDisplayRes()` and `horizontalDisplayRes()` are not in TB30DOS.DLL as stated on this page, but in TB30WIN.DLL.

Finding strings with the `is in` operator

Where: ToolBook User Manual, Appendix A, page A-11

The `in` operator in the fifth bulleted paragraph should be `is in`.

`size` property

Where: OpenScript Reference Manual, page 1-4 and following

In all references to the `size` property, the description should note that the property returns the "width and height" of the object, not the other way around. The first value in the `size` property is the object's width; the second is the object's height.

`readerStatusBar` viewer property

Where: OpenScript Reference Manual, page 1-54

The entry for `readerStatusBar` on this page incorrectly states that the property specifies whether the status bar can be visible in a viewer at Reader level. It should state that this property determines whether a status bar is visible when the viewer first appears. The entry for this property on page 2-242 is correct.

`autoSize` viewer property

Where: OpenScript Reference Manual, page 2-23

The documentation for a viewer's `autoSize` property states that if this property is `true`, the viewer resizes itself to its current page each time the viewer is shown or its `currentPage` property changes. More correctly, the viewer resizes itself any time the page size changes, either because the `currentPage` changes to a page of a different size, or because the background or book `pageSize` property defining the size of the `currentPage` changes.

`clear` command

Where: OpenScript Reference Manual, page 2-47

The description of the `clear` command should note that if you attempt to clear a property that requires a value, Multimedia ToolBook displays an error. For example, the following command produces an error:

```
clear saveOnClose of this book
```

`createHotword` message

Where: OpenScript Reference Manual, page 2-62

This message is incorrectly listed as causing an error in Runtime Multimedia ToolBook; the message works as documented in both the full and Runtime versions of Multimedia ToolBook.

`defaultPosition` of viewers

Where: OpenScript Reference Manual, page 2-70

The default position of viewers is not `center`, as documented under `defaultPosition`, but `none`. This applies to viewers you create as well as to the Main window.

`GDIHandle()` function

Where: OpenScript Reference Manual, page 2-122

This function cannot be used to return the handle for a menu resource.

International system properties

Where: OpenScript Reference Manual, page 2-145, 2-149

The properties `sysIDigits` and `sysILZero` do not automatically set the default value of `sysNumberFormat`. You can set a new value for `sysNumberFormat` using explicit formats, or by concatenating explicit values with the values of `sysIDigits` and `sysILZero`.

`parentHandle` property

Where: OpenScript Reference Manual, page 2-216

The second paragraph states that valid values for this property are window handles or `null`. It should state that valid values are window handles or zero (0).

`richText` property

Where: OpenScript Reference Manual, page 2-263

The second paragraph states that if you extract a smaller section of rich text from another field, the `richText` property returns `null`. It should state that the property returns the requested value based on the value of the field including the RTF information. For example, if you request `textline 2 of richText of field "test"`, Multimedia ToolBook returns whatever string of characters is found between the first and second carriage return-linefeed combinations. That information will almost certainly be from the RTF header information instead of from the text.

Because it is almost always impossible to predict what information will be returned in this way, it is recommended that you not attempt to extract substrings using the `richText` property.

`seekFile` command

Where: OpenScript Reference Manual, page 2-280

The syntax for the seekFile command incorrectly uses the word `to` instead of `for`. The syntax statement should read:

```
seekfile <file name> for <position> from <location>
```

The examples on the following page are correct.

`storedImages` property

Where: OpenScript Reference Manual, page 2-315

The description of the value returned by this property is not correct. The documentation should state that the property returns a series of textlines, each of which contains five items:

Item	Contents	Description
1	bits per pixel	Number of bits per pixel used by the display device
2	color planes	Number of color planes used by the display device
3	logical x units	Number of logical horizontal units used by the display device
4	logical y units	Number of logical vertical units used by the display device
5	size	Size of the stored image in kilobytes

`targetWindow` property

Where: OpenScript Reference Manual, page 2-347, 2-348

On page 2-347: The commands `print`, `print eject`, and `sort` are not available in the context of the target window.

On page 2-348: The last line of the last example has an extra set of quotation marks around the page reference. The line should read:

```
currentPage of targetWindow = page 1 of book "bar.tbk"
```

Style of target

Where: OpenScript Reference Manual, page 2-356

In the third line of the example, the property should not be `buttonStyle`, but `borderStyle`. The correct example is:

```
return the borderStyle of the target
```

`transition` command

Where: OpenScript Reference Manual, page 2-362

The transition effect `slide` does not support a direction (`in` or `out`).

`getListBoxSelection()` function

Where: OpenScript Reference Manual, page 3-19

The syntax for this function lists a parameter called `<selection>` that does not exist. The declaration and example are correct.

`getModuleList()`, `getModulePath()`

Where: OpenScript Reference Manual, pages 3-55, 3-56

The example programs for the DLL functions `getModuleList()` and `getModulePath()` link the incorrect DLL. Both functions are in TB30WIN.DLL, so the correct form of the `linkDLL` statement is:

```
linkDLL "TB30WIN.DLL"
```

`textFromPoint()` function

Where: OpenScript Reference Manual, page A-14 in the Appendix

The `textFromPoint()` function does not accept a second parameter that refers to a viewer, as suggested by the entry under "Mouse and keyboard functions." The function always operates in the context of the target viewer.

`sysError` at idle

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 6-17

Multimedia ToolBook does not set `sysError` to `true` when it reaches its idle state; you must manually set this property to prevent errors from compounding.

`CDMediaPath` in comment

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 6-31

In the last example on the page, the first comment references a property called `sysCDMediaPath`, which doesn't exist. The line should read:

```
--Sets CDMediaPath to first CD-ROM drive and
```

The error does not affect how the example runs, but could be confusing.

`HDMediaPath` property

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 10-9

In the third line of the example for `HDMediaPath`, the phrase `of this book` is missing. The line should read:

```
push "c:\media\sound\,c:\media\video" onto HDMediaPath of this book
```

`mmBackgroundPalette` property

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 10-17

The default for the `mmBackgroundPalette` property is `true`, not `false`.

`mmClose` command

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 10-18

If you use specify a particular media type rather than a media clip, the media type must be in quotation marks. For example, to close all digital audio clips, use a command such as this:

```
mmClose "waveAudio"
```

`mmPause` command

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 10-24

If you use specify a particular media type rather than a media clip, the media type must be in quotation marks. For example, to pause all digital audio clips, use a command such as this:

```
mmPause "waveAudio"
```

`mmRewind` command

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 10-28

If you use specify a particular media type rather than a media clip, the media type must be in quotation marks. For example, to rewind all digital audio clips, use a command such as this:

```
mmRewind "waveAudio"
```

`mmStatus` property

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 10-34

The description of the `mmStatus` property incorrectly states that it is nonpersistent. The property is persistent; you can get the value of this property for a clip that is closed.

`mmStop` command

Where: Multimedia ToolBook User Manual and OpenScript Reference, page 10-36

If you use specify a particular media type rather than a media clip, the media type must be in quotation marks. For example, to stop all digital audio clips, use a command such as this:

```
mmStop "waveAudio"
```

6.0. Technical support contact information

Telephone support

Contact Asymetrix at the telephone numbers listed below for information on telephone support contracts. Registered ToolBook 3.0 and Multimedia ToolBook 3.0 users receive 60 days of complimentary technical support, beginning with the first call.

Australia/Asia Pacific	(61+3) 5255471
Europe (except France and Germany), Middle East, Africa, Russia	0923-420-54-33
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 to registered users via fax, BBS, CompuServe, America Online, and Internet. Technical support responds to online queries within 48 hours (Monday to Friday).

Technical support fax

* Australia/Asia Pacific	(61+3) 5255-482
* 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