



macromedia
AUTHORWARE 7

Using Authorware 7



Trademarks

Afterburner, AppletAce, Attain, Attain Enterprise Learning System, Attain Essentials, Attain Objects for Dreamweaver, Authorware, Authorware Attain, Authorware Interactive Studio, Authorware Star, Authorware Synergy, Backstage, Backstage Designer, Backstage Desktop Studio, Backstage Enterprise Studio, Backstage Internet Studio, Contribute, Design in Motion, Director, Director Multimedia Studio, Doc Around the Clock, Dreamweaver, Dreamweaver Attain, Drumbear, Drumbear 2000, Extreme 3D, Fireworks, Flash, Fontographer, FreeHand, FreeHand Graphics Studio, Generator, Generator Developer's Studio, Generator Dynamic Graphics Server, Knowledge Objects, Knowledge Stream, Knowledge Track, LikeMinds, Lingo, Live Effects, MacRecorder Logo and Design, Macromedia, Macromedia Contribute, Macromedia Coursebuilder for Dreamweaver, Macromedia M Logo & Design, Macromedia Flash, Macromedia Xres, Macromind, Macromind Action, MAGIC, Mediamaker, Multimedia is the Message, Object Authoring, Power Applets, Priority Access, Roundtrip HTML, Scriptlets, SoundEdit, ShockRave, Shockmachine, Shockwave, shockwave.com, Shockwave Remote, Shockwave Internet Studio, Showcase, Tools to Power Your Ideas, Universal Media, Virtuoso, Web Design 101, Whirlwind and Xtra are trademarks of Macromedia, Inc. and may be registered in the United States or in other jurisdictions including internationally. Other product names, logos, designs, titles, words or phrases mentioned within this publication may be trademarks, servicemarks, or tradenames of Macromedia, Inc. or other entities and may be registered in certain jurisdictions including internationally.

This guide contains links to third-party Web sites that are not under the control of Macromedia, and Macromedia is not responsible for the content on any linked site. If you access a third-party Web site mentioned in this guide, then you do so at your own risk. Macromedia provides these links only as a convenience, and the inclusion of the link does not imply that Macromedia endorses or accepts any responsibility for the content on those third-party sites.

Apple Disclaimer

APPLE COMPUTER, INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED COMPUTER SOFTWARE PACKAGE, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY PROVIDES YOU WITH SPECIFIC LEGAL RIGHTS. THERE MAY BE OTHER RIGHTS THAT YOU MAY HAVE WHICH VARY FROM STATE TO STATE.

Copyright © 2003 Macromedia, Inc. All rights reserved. This manual may not be copied, photocopied, reproduced, translated, or converted to any electronic or machine-readable form in whole or in part without prior written approval of Macromedia, Inc. Part Number ZAP70M100

Acknowledgments

Project Management: Gary White

Writing: Naheeda Ravjani

Editorial Managment: Rosana Francescato

Editing: Noreen Maher, Mary Ferguson, Mary Kraemer, Lisa Stanziano

Production Management: Patrice O'Neill

Multimedia Design and Production: Aaron Begley

Production: Adam Barnett, John Francis

Special thanks to

Jamil Zainasheff, Tom Neuhold-Huber, Chris Dant, Jon Violette, Steve Gannon, Stefan Van As, Joe Ganci, Chris Maier, Mike Baker, Miriam Geller, Grace Gellerman.

First Edition: May 2003

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

CONTENTS

INTRODUCTION: Introducing Authorware 7	13
System requirements	14
Installing Authorware 7	14
If you're new to Authorware	14
 CHAPTER 1: What's New in Authorware 7	17
Interface improvements	17
Accessibility	17
New media features	17
JavaScript engine	18
PowerPoint to Authorware	18
Trace/Debug window improvements	18
Standards and tracking	18
New properties, functions, and variables	18
New functions	18
Open properties	20
New variables	21
 CHAPTER 2: Authoring Basics	23
Icon-based authoring	23
The stages of authoring	24
Authoring—step-by-step procedures	25
The toolbar	27
The Icon palette	27
The toolbox	27
The Tools panel	27
The Property inspector	27
Icon basics	28
Inserting an icon	29
Setting up icons	29
Setting up a display icon	30
Setting up a motion icon	30
Setting up an erase icon	30
Setting up a wait icon	31
Setting up a navigate icon	31
Setting up a framework icon	31

Setting up a decision icon	32
Setting up an interaction icon	32
Setting up a calculation icon	32
Setting up a map icon	33
Setting up a digital movie icon	33
Setting up a sound icon	34
Setting up a DVD icon	34
Setting up a sprite icon	34
Tips for working with icons	35
Editing an icon	36
Editing multiple icons	36
Displaying the content of several icons	37
Displaying the Design window	37
Using Knowledge Objects	37
Selecting a Knowledge Object for a new file	37
Adding a Knowledge Object to a file	38
Authorware Knowledge Objects	38
New File Knowledge Objects	38
Assessment Knowledge Objects	39
File Knowledge Objects	40
Interface Components Knowledge Objects	40
Internet Knowledge Objects	41
RTF Objects Knowledge Objects	41
CHAPTER 3: Working with Authorware	43
Good authoring practice	43
Troubleshooting methods	45
Troubleshooting—step-by-step procedures	47
Setting up file structures	47
Default search path	48
Specifying file locations	48
User records	49
Location of user records	49
Changing the records folder	50
Specifying where Authorware searches for files	50
Distributing your piece	50
Selecting a distribution medium	51
Packaging an Authorware piece	52
Options for packaging a piece	52
Packaging a piece—step-by-step procedures	54
Getting a piece ready for the web	54
Getting ready for the web—step-by-step procedures	55
Using one-button publishing to package a piece	55
Setting up one-button publishing settings	55
Determining which files you need to distribute	56
Commands menu	57
Adding your own commands to the Commands menu	58

CHAPTER 4: Working with Content	59
Creating and incorporating content	59
Embedding and linking content	60
Embedding and linking content—step-by-step procedures	60
Working with text	61
Guidelines for creating and integrating text	61
Creating and integrating text	62
Making the most of text	63
Controlling text with variables and functions	63
Text—step-by-step procedures	65
Using the RTF Objects Editor	66
Definitions	66
Features	67
Opening the RTF Objects Editor	67
The RTF Objects Editor toolbar	68
Managing files with the RTF Objects Editor	68
Opening a new RTF document	68
Opening an existing RTF document	68
Opening a database	69
Opening a previously opened RTF document	70
Saving an RTF document	70
Printing an RTF document	70
Exiting the RTF Objects Editor	71
Editing files with the RTF Objects Editor	71
Copying text or graphics	72
Pasting text or graphics	72
Using the Paste Special command	73
Searching for and replacing text in the RTF Objects Editor	73
Changing view options in the RTF Objects Editor	75
Inserting files and graphics using the RTF Objects Editor	75
Inserting a file or image into an RTF document	75
Inserting a shape in an RTF document	76
Inserting a symbol in an RTF document	77
Inserting a hyperlink (hot text) in an RTF document	77
Inserting a page break in an RTF document	78
Inserting the date and time in an RTF document	79
Inserting an Authorware expression in an RTF document	79
Changing formats with the RTF Objects Editor	80
Using RTF Knowledge Objects	81
Creating an RTF Object Knowledge Object	82
Exporting an RTF Object Knowledge Object	82
Using the Show or Hide RTF Object Knowledge Object	82
Using the Get RTF Object Text Range Knowledge Object	82
Inserting an RTF Object Hot Text Interaction Knowledge Object	82
Searching for text in an RTF Object Knowledge Object	83
Using RTF functions	83
RTF functions	84
Working with images and graphic objects	87
Creating and integrating images and graphics	88

Making the most of images and graphics	89
Controlling images with variables and functions	90
Graphics—step-by-step procedures	91
Working with sound	92
File formats for sounds	93
Creating and integrating sound	93
Synchronizing audio, DVDs, and graphics	93
Media synchronization	94
Controlling sound with variables and functions.	94
Sound—step-by-step procedures	95
Working with digital movies	95
Digital movie file formats	96
Creating and integrating digital movies	96
Making the most of digital movies.	97
Variables and functions for digital movies	98
Digital movies—step-by-step procedures	98
Working with DVDs.	99
Creating and integrating DVDs.	99
Controlling DVDs with variables and functions	100
DVDs—step-by-step procedures	102
Managing content	102
External content, libraries, and models	103
Using external content and the External Media Browser	105
External Media Browser—step-by-step procedures	106
Using libraries	106
Libraries—step-by-step procedures	108
Using models.	108
Models—step-by-step procedures	109
CHAPTER 5: Creating Interactions	111
The components of an interaction.	111
How an interaction works	112
Tracing the flow through an interaction.	114
Ways to interact.	116
Setting up an interaction—step-by-step procedures.	117
Buttons, radio buttons, and check boxes	118
Buttons—step-by-step procedures	119
Pull-down menus	119
Pull-down menus—step-by-step procedures	120
Hot spot interactions.	120
Hot spots—step-by-step procedures	121
Hot object interactions	121
Hot objects—step-by-step procedures	122
Keypress interactions	122
Keypress responses—step-by-step procedures	123
Text entry interactions.	123
Text entry—step-by-step procedures	123
Target area interactions	124
Target areas—step-by-step procedures	124

Tries limit—limiting users' attempts	125
Tries limit—step-by-step procedures	126
Time limit interactions	126
Time limit—step-by-step procedures.	126
Conditional interactions	126
Conditional responses—step-by-step procedures	128
Events and senders.	128
Events—step-by-step procedures	129
Perpetual interactions—step-by-step procedures	129
Limiting when a choice is active	130
Limiting when a choice is active—step-by-step procedures	130
CHAPTER 6: Directing the Flow	131
Ways of directing the flow	131
User control versus automatic control.	131
Following a path versus jumping around	132
The flow within a file versus the flow between files.	132
The flow and the Internet.	132
Navigate, framework, interaction, and decision icons	133
Decision structures—directing the flow automatically.	133
Types of branching	134
Sequential branching.	134
Random branching	135
Branching based on a variable or expression	135
Branching based on a calculation	136
Branching based on a conditional expression	136
Decision structures—step-by-step procedures	137
Paging structures—quick and easy user control	137
Paging structures—step-by-step procedures	138
Frameworks in depth.	139
How the framework icon works.	140
The framework's navigation controls	141
Nesting navigation frameworks	142
Things to consider about navigation structures	142
Navigation structures—step-by-step procedures	143
Navigation links—jumping from place to place.	144
Anywhere—going to a specific page	144
Recent—backtracking	144
Nearby—paging	144
Calculate—using an expression to go to a page.	145
Search—allowing users to search for a page	145
One-way trips and round trips	146
Nesting calls	147
Navigation links—step-by-step procedures	147
Navigating with hypertext.	148
Hypertext—step-by-step procedures.	148
Jumping to another file	148
Subroutines	149

CHAPTER 7: Transitions, Positioning, and Motion	151
Using transitions for special effects	151
Transition settings.	152
Transition Xtra extensions	152
Positioning objects.	152
Positioning objects on a path.	153
Positioning objects within an area	153
Positioning objects on the screen	153
Making objects move.	154
Using the motion icon.	154
Letting users drag objects.	155
Variables for tracking an object's motion and position	155
Making objects move—step-by-step procedures	156
 CHAPTER 8: Using Variables, Functions, Xtra Extensions, and More	 157
What each variable and function does	158
Where you can use variables, functions, expressions, code snippets, and scripts	158
Using the Calculation window	160
Changing calculation defaults in the Calculation window.	161
Inserting a message box, snippet, or symbol into a calculation	161
Displaying the ASCII value of selected text.	161
Calculation window keyboard shortcuts	161
Creating a Quit button.	164
Variables and Functions dialog boxes.	165
Using comments	165
Variable and function basics—step-by-step procedures	165
Using system and custom variables	165
System variables.	166
Custom variables	166
Types of variables.	166
Variables—step-by-step procedures	168
Functions	168
System functions	168
Getting the syntax right.	169
Custom functions in DLLs	169
Loading a DLL custom function	170
Using functions—step-by-step procedures.	171
Expressions and operators	171
Types of operators	171
Code Snippets	173
Scripting	174
If...then statements	175
Repeat loops	176
Scripting—step-by-step procedures	177
Xtra extensions	177
Types of Xtra extensions	177
Sprite Xtra extensions	178

Sprite properties, events, and methods	179
Sprite properties	179
Sprite events	180
Sprite methods	180
Scripting Xtra extensions	181
Global functions, parent objects, and child objects	181
Using parent and child objects	181
Functions for using parent and child object methods	182
Identifying global functions and parent and child methods.	182
ActiveX controls	182
ActiveX controls—step-by-step procedures	183
The XMLParser Xtra	183
About XML	183
 CHAPTER 9: Tracking Performance and Working with a	
Learning Management System	185
Using Assessment Knowledge Objects	185
QTI Assessment Knowledge Objects	185
Authorware Learning Object Content Packager command	186
Authorware Learning Object Metadata Editor	186
Guidelines for tracking and recording performance	186
Marking a response right or wrong	187
Tracking performance	188
Providing feedback	190
Recording performance	191
Storing information in Authorware.	191
Saving information in an external file	192
Saving information on a local hard disk or server	192
Saving information over the Internet	192
Managing information with a database	192
Tracking and recording performance—step-by-step procedures	193
Using Knowledge Track to set up a course	193
Knowledge Track settings	193
CMI variables and functions	194
Learning Management System (LMS) examples	198
Connecting with databases	200
 CHAPTER 10: Streaming, Intranets, and the Web	201
Speed.	201
Streaming technology for the web	202
Designing a piece for the web	203
Tips for optimizing web-packaged pieces for low-bandwidth (28.8 Kbps) delivery	204
Web-related variables and functions	206
Variables	206
Managing content in web-packaged pieces	208
Configuring your server.	208
Providing the Authorware Web Player	209

Streaming—step-by-step procedures	209
Authorware Advanced Streamer.	210
Using Internet services.	210
Designing for Internet services	210
The Authorware FTP functions.	211
CD—web hybrids	212
 CHAPTER 11: Creating Knowledge Objects	 213
Displaying Knowledge Objects in the Knowledge Objects window	214
Wizards and targets	214
Variables and functions for Knowledge Objects	215
Knowledge Objects—step-by-step procedures	219
The Knowledge Object icon	220
Knowledge Object icon options.	220
Creating a good wizard user interface	220
Communicating with a Knowledge Object	221
 CHAPTER 12: Show Me Examples	 223
Accessing Show Me examples	223
24 Points (24points.a7p)	223
Administration (Admin.a7p)	223
Altimeter (Altimeter.a7p)	223
Animated Buttons (anibutt.a7p)	223
A/V Synchronization (Avsync.a7p)	223
Bar Graph (Bargraph.a7p)	223
Button Interaction (Q_button.a7p)	224
Calculator (Calculator.a7p)	224
Catch It (CatchIt.a7p)	224
CD-ROM Drive Letter (FindCD.a7p)	224
Clock (Clock.a7p)	224
Content Application (Content.a7p)	224
Creating a Subroutine (Subrout.a7p)	224
Cubes (Cubes.a7p)	224
Displaying Variables (Variable.a7p)	224
Double Click Selection (Dbldclick.a7p)	224
Drag and Drop Interactions (Drag.a7p)	224
Encryption-Decryption (Encrypt.a7p)	224
External Content (External\External.a7p)	224
Flash in ActiveX (flash_ax\FlashAx.a7p)	225
Forms (Forms.a7p)	225
Forms with WinCtrls (Forms_Win.a7p)	225
Free Form Text (Q_txt_1.a7p)	225
Generic Question Model (Q_model.a7p)	225
Glossary (Glossary.a7p)	225
Graphing (Graph.a7p)	225
Hangman (Hangman.a7p)	225
Help System (Help.a7p)	225
Hot Object Interaction (Q_hotobj.a7p)	225

Hot Spot Interaction (Q_hotspt.a7p)	225
Hot Text (Hottext.a7p)	225
Jigsaw Puzzle (Jigsaw.a7p).	226
Judging and Tracking (Judge.a7p).	226
Keyboard (Keyboard.a7p).	226
Keypress Interactions (Keypress.a7p)	226
Logon (Logon.a7p).	226
Magnify (Magnify.a7p).	226
Making Icons Immovable (Lockdown.a7p).	226
Marching Ants (Marchant.a7p)	226
Memory Game (Memgame.a7p)	226
Morse Code (Morse.a7p)	226
Movie Controls (Movctrls.a7p).	226
Moving Objects (Move.a7p).	227
Mr. ASCII Pseudo (Ascii.a7p).	227
Navigating & Expressions (Nav_page.a7p)	227
Navigating with Flash (flash_nav\Nav_Flash.a7p)	227
Notepad (Notepad.a7p)	227
Orbit (Orbit.a7p)	227
Paging (Paging.a7p)	227
Picture Box (Pic-box.a7p).	227
Pie Chart (PieChart.a7p)	227
Piston (Piston.a7p)	227
Pop-up Glossary (Popup.a7p)	227
Radio Buttons (Radiob.a7p).	228
Rat Maze (Ratmaze.a7p).	228
RealNetworks (Real.a7p)	228
Referring to Icons (Refer.a7p).	228
RGB (RGB.a7p).	228
Right and Left Clicks (Rightclk.a7p)	228
Rotating Graphic (Rotate.a7p)	228
RTF Objects Editor (RTF\RTFa7p).	228
Screen Capture (screencp\Screencp.a7p).	228
Screen Saver (Ssaver.a7p)	228
Screening Text or Numbers (Screen.a7p)	228
Scrollable Lists (Scrlist.a7p)	228
Scrolling Checklist (Scrollck.a7p).	229
Set Motion Object (setmotionobject.a7p).	229
Short Text Interaction (Q_txt_s.a7p)	229
Show Me Credits (Credits.a7p).	229
Slider (Slider.a7p).	229
Smart Glossary (Smart.a7p)	229
Snake (Snake.a7p)	229
Storing/Retrieving Information (Write.a7p)	229
Tab Buttons (Tabs.a7p)	229
Target Area Interaction (Q_target.a7p).	229
Ticker Tape (Ticker.a7p)	229
Time Limit Interaction (Q_time.a7p)	229
Tool Palette (Toolpal.a7p).	229
Tool Tips (Tooltips.a7p)	230

Tries Limit Interaction (Q_tries.a7p)	230
Using ActiveX (activex\Activex.a7p)	230
Using Lists (Lists.a7p)	230
Using ODBC (Odbc\Odbc.a7p)	230
Using Scripting Xtras (Xtraio.a7p)	230
Variable Branching (Decision.a7p)	230
Variable Button Labels (vaributt.a7p)	230
Window Resize (Winsize.a7p)	230
XML Parser (XMLParser.a7p)	230
 CHAPTER 13: Using Windows Controls Knowledge Objects and WinCtrls.u32	231
About WinCtrls.u32.	231
About Windows Control Knowledge Objects	232
Using a Windows Control Knowledge Object to create a Windows control . .	233
Setting a control's properties with the Set Property Knowledge Object	236
Retrieving a control's properties with the Get Property Knowledge Object . .	237
Example: Building a list box	240
Available Windows controls and properties	243
Using Authorware Windows controls variables	245
Using the change event variable name.	246
Using List Variables	246
About the control ID number.	246
Coding a control	247
Using user icon variables.	248
 CHAPTER 14: Using JavaScript in Authorware	251
About JavaScript	251
Authorware Document Object Model	252
Authorware JavaScript language extensions	253
Defining Authorware.	253
Entering JavaScript code	253
JavaScript Variables	254
Execution speed.	255
The Calculation editor and languages	255
The aw object	255
Case sensitivity	257
Duplicate system functions and variables.	257
Icon object	258
New Authorware functions	259
Enhanced Authorware functions	261
Helpful hints for using JavaScript	262
 INDEX	269

INTRODUCTION

Introducing Authorware 7

Macromedia Authorware 7 is the leading rich-media authoring tool for creating web and online learning. Authorware integrates graphics, sound, animation, text, and video into compelling rich-media learning solutions.

Authorware is optimized for building e-learning applications. The Authorware interface provides a fast and easy environment for creating interactive applications. Drag-and-drop icons support rapid application prototyping and development without the need for extensive scripting. The flowline provides an intuitive representation of the structure of large, highly branched learning applications.

Authorware 7 has been updated with a new interface. The new look includes panels that dock on the right side of the application window and a new Property inspector that significantly streamlines workflow.

Scripting in Authorware is now possible in JavaScript as well as Authorware Script Language. The Calculation editor lets you specify either language for entering and running scripts. The Preferences dialog box now lets you specify the appearance of script in both languages.

A major addition to Authorware 7 is XML import and export capability. Not only can you import XML files into Authorware but you can export an entire Authorware piece as XML. For example, you can import a Microsoft PowerPoint presentation as XML into Authorware, and use the PowerPoint slides as the foundation for building an Authorware application.

Authorware now supports DVD. You can set up Authorware to automatically start and stop at specific points within a DVD movie, and give students playback controls to play, pause, or replay movie segments.

Improvements to Trace/Debug include new Stop, Start, and Wait commands to give you more control over debugging.

Authorware 7 also includes support for Mac OS X playback as well as ADL SCORM API Adapter Objects. Additionally, the new Learning Object Packager creates SCORM-compliant learning objects from Authorware files.

Authorware 7 is part of the eLearning studio group of integrated, open, and scalable learning tools and technologies. This system provides a complete e-learning solution, including the creation, delivery, and management of enterprise learning programs.

System requirements

Component	Authoring	Playback
Processor	Intel Pentium II or higher	Intel Pentium II or higher
Memory	32MB RAM	16MB RAM (Microsoft Windows) 24MB RAM (Mac)
System software	Windows XP, Windows 2000, Windows 98SE	Windows XP, Windows 2000, Windows NT 4.0, Windows Me, Windows 98SE Mac OS 8.1 through OS X
Drive	120MB of free disk space and a CD-ROM drive	Not applicable

Installing Authorware 7

To install Authorware 7:

- 1 Put the Authorware 7 CD into your CD-ROM drive.
- 2 Double-click the Setup icon.
- 3 Follow the prompts to customize the installation and select file designations.

When installation is complete, be sure to read the Authorware 7 Read Me file. It contains late-breaking information.

If you're new to Authorware

Authorware is a sophisticated tool that's both powerful and easy to use. You'll learn the basics quickly; other things will take you more time to master. This checklist will help you find the resources you need to learn everything you need to know about using Authorware.

- Use the Macromedia Knowledge Objects.

Knowledge Objects have wizards that lead you through the steps of creating projects in Authorware. Launch Authorware, and then select a Knowledge Object from the Knowledge Object window in the Tools panel. Drag the Knowledge Objects you need from the Knowledge Objects window.

- Work through the online tutorial.

The tutorial introduces the basics of using Authorware. You can find it at www.macromedia.com/support/authorware, or select Help > Support Center.

- Read *Using Authorware 7*.

The Authorware 7 manual is the best place to learn about the concepts behind Authorware. Don't overlook the step-by-step tables that tell you where to look for information in Authorware Help. For information on new features that have been added to Authorware 7, read [Chapter 1, "What's New in Authorware 7," on page 17](#). To get started, read [Chapter 2, "Authoring Basics," on page 23](#).

- Use Authorware Help.

Online help is available directly in Authorware Help. Select Help > Authorware Help. The help system contains procedures, detailed descriptions of functions and variables, and reference information.

- Look at the Show Me examples.

Authorware comes with 80 Show Me examples that not only answer questions but also provide flowline logic that you can paste into your work. Install the Show Me examples on your hard disk for quick access from the Help menu (select Help > Show Me). More Show Me examples are added to the Authorware Support Center every month (select Help > Support Center or visit www.macromedia.com/support/authorware).

- Visit the Authorware Support Center.

The Authorware Support Center is a rich source of information about Authorware that includes articles by expert users on new features and common problems, regular installments of new Show Me examples, and information about books and websites, user groups and events, and tools and services you may need. Go to the Authorware Support Center at www.macromedia.com/support/authorware or select Help > Support Center.

- Join the AWARE list.

The AWARE discussion list is an online forum for discussing Authorware. This list is not maintained by Macromedia; it is maintained by Authorware developers worldwide. You can subscribe to the list (you'll receive around 1000 e-mail messages a month), or you can browse the newsgroups bit.listserv.authorware and alt.authorware.

To subscribe to the AWARE discussion list, send the following message to LISTSERV@LISTSERV.CC.KULEUVEN.AC.BE

SUBSCRIBE AWARE <Your Name>

For example, **SUBSCRIBE AWARE Chris Chen**

You can remove your name from the list at any time by sending a `SIGNOFF AWARE` or `UNSUBSCRIBE AWARE` command to the listserv address. For more information on the AWARE list, visit <http://ls.kuleuven.ac.be/archives/aware.html>.

- Check the Authorware TechNotes when you have a problem.

If you have a problem, chances are that Technical Support has already solved the same problem for someone else. See the TechNote Index on the Authorware Support Center at www.macromedia.com/support/authorware/technotes.html or select Help > TechNotes.

- Check the Authorware discussion group.

If you have a problem and can't find the answer in the TechNotes, check the Authorware discussion group. The discussion group is part of the Authorware Support Center at www.macromedia.com/support/authorware or select Help > Support Center.

- Buy a book.

Macromedia Press and other publishers publish books by independent authors. You might find one or more of them helpful. Macromedia Press books are published by Peachpit Press. For information on these, call 800-283-9444 or visit www.peachpit.com.

- Join a user group.

Other developers can help you over a lot of hurdles. The best place to meet them is at a user group. For a list of user groups, visit the Authorware Support Center at www.macromedia.com/support/authorware or select Help > Support Center.

CHAPTER 1

What's New in Authorware 7

Macromedia Authorware 7 provides all the tools you need to create media-rich, interactive, and trackable e-learning for delivery on many platforms, including the web, corporate networks, and CDs.

The following sections give an overview of the new features in Authorware 7.

Interface improvements

Authorware has a modern appearance, to provide an enhanced user experience:

- It's easier to change icon properties with the new Property inspector.
- The Toolbar and Icon palette are movable.
- Knowledge Objects, functions, and variables are easily accessible in panels that dock to the right side of the application window.
- The Preferences dialog box is easier to use, with more Calculation window buttons enabled by default and specific Authorware and JavaScript (JS) tabs.

Accessibility

Authorware now has a new Speech Xtra extension that allows easier implementation of text-to-speech on a range of end-user computers, including the Macintosh and web player in trusted mode.

New media features

Authorware users can now integrate DVD movies as components of their training applications. The Laserdisc Movie icon has been replaced with a DVD Movie icon. The former Video functions have been replaced by DVD functions. The DVD functions are located within the Video category in the Functions panel.

In addition, Authorware users now have a stand-alone runtime player, packaging tool, and web player for Mac OS X environments.

JavaScript engine

Authorware now supports coding and executing JavaScript. You have a new choice of programming languages to use within calculation icons. You can continue to use the original language (Authorware Script Language, or AWS), or you can use the new support for JavaScript, version 1.5.

PowerPoint to Authorware

You can now import Microsoft PowerPoint presentations into Authorware. Authorware converts a PowerPoint file into XML and then imports it directly onto the Authorware flowline. Then you can add sophisticated branching, simulations, assessments, and a host of other features. This process lets you quickly and easily create powerful, media rich, e-learning modules.

Trace/Debug window improvements

Authorware users now have additional features to help with the debugging process. The user can specify which variables are visible, and can step through calculation icons.

Standards and tracking

Authorware has a new Learning Management System (LMS) Knowledge Object (KO). This new KO lets Authorware communicate with any AICC/SCORM-compliant LMS. The LMS KO contains a wizard that guides the user through the once-daunting task of setting up communication with an LMS.

New properties, functions, and variables

Many new properties, functions, and variables are provided with Authorware 7. For complete details, see System functions in Authorware Help.

New functions

New functions have been added to Authorware 7. For complete details, see System functions in Authorware Help.

Function	Description
BuildDisplay	Adds a new object to the display specified in IconID, and describes the object in the ObjectList argument. If a third argument is included, the item is inserted into existing items at the location specified by index.
CMIShowErrors	Controls whether error dialog boxes automatically appear to the user at runtime. The default is to show errors. If show is true, errors appear. If show is false, errors do not appear.
DVDAction	Affects DVD playback based on the symbol specified.
DVDCaptions	Turns on the DVD captions when TRUE (if they are present). Specify FALSE to turn captions off.
DVDChapterNum	Returns the number of the current chapter.

Function	Description
DVDCreate	Creates the DVD playback window at the window positions and sizes specified. <code>DVDFileName</code> is the full drive letter, path, and filename of the file to open when creating the DVD window. All the parameters are optional. If you do not specify both the <code>WindowLeft</code> and <code>WindowTop</code> (or they are zero), the window origin is at 0, 0. If you do not specify the <code>WindowHeight</code> and <code>WindowWidth</code> (or they are zero), it creates the DVD window at a size to match the current size of the Authorware Presentation window. If you omit the <code>DVDFileName</code> (or pass a null string), <code>DVDCreate</code> finds the first DVD disc in a DVD drive on the system and attempts to load that disc. The initial window is created hidden, to allow for audio-only playback. To display the DVD window, use <code>DVDShowWindow(TRUE)</code> . Returns true if successful.
DVDCurrentTitleNum	Returns the current title number.
DVDDirectX	Returns the version number of Microsoft Windows DirectX support installed.
DVDGetDrive	Returns the drive letter of the first DVD drive it finds on the system with a DVD disc loaded. <code>DVDGetDrive</code> begins its search from the drive letter specified. Returns a null string if no DVD drive with a disc is found.
DVDGetVolume	Returns the DVD audio level as a percentage: 100 is full volume, and 0 is no volume.
DVDMute	Mutes the DVD audio when TRUE. Specify FALSE to restore the sound.
DVDNumChapters	Returns the number of chapters in the specified title. If <code>TitleNumber</code> is omitted, it returns the number of chapters in the current title.
DVDNumTitles	Returns the number of titles on the current DVD.
DVDPlayChapter	Plays the specified chapter in the specified title. If <code>TitleNumber</code> is 0, the chapter is played from the current title.
DVDPlaytime	Plays the specified Title number from the specified start time to the specified end time. If <code>Title</code> is 0, the selection plays from the current title number. If the end time is omitted, the playback continues to the end of the title.
DVDSelectButton	Selects the specified button number from the DVD menu.
DVDSetVolume	Sets the DVD audio level as a percentage: 100 is full volume, and 0 is no volume.
DVDShowWindow	Creates the DVD window in a hidden state. Use <code>DVDShowWindow(TRUE)</code> to show the window. Use <code>DVDShowWindow(FALSE)</code> to hide the DVD window if it is visible.
EvalJS	Executes the JavaScript string and returns the result. The JavaScript is evaluated in the context of Authorware. There are no browser window objects. If a syntax error occurs, it is noted in <code>EvalStatus</code> and <code>EvalMessage</code> .
EvalJSFile	Reads the specified file, executes the string as JavaScript, and returns the result. The JavaScript is evaluated in the context of Authorware. There are no browser window objects. If a syntax error occurs, it is noted in <code>EvalStatus</code> and <code>EvalMessage</code> .
InsertIcon	Inserts an icon of type "IconType" on the Authorware flow line at the current location of the paste hand.

Function	Description
<code>JSGarbageCollect</code>	Performs garbage collection in the JavaScript memory pool. Garbage collection, if necessary, frees the memory used by JavaScript objects, doubles, and strings that are no longer needed by a script that is executing in the current context. Garbage collection frees space in the memory pool so that it can be reused by the JavaScript engine. Generally, garbage collection is best handled automatically by the JavaScript engine and this function does not need to be called.
<code>ShowTitleBar</code>	On the Macintosh, <code>ShowTitleBar</code> turns the presentation window title bar on or off. Specify OFF as the argument to hide the title bar; specify ON to display the title bar. In Windows and on the Macintosh, you can use the File Properties dialog box to turn the title bar on or off for the entire presentation. Use <code>ShowTitleBar</code> only in calculation icons.

Open properties

The following are the new properties available for `GetIconProperty` and `SetIconProperty`.

Authorware name	Property	Icons
JavaScript Calculation	<code>#awCalcIsJavascript</code>	Calculation
Video Filename	<code>#awDVDFilename</code>	DVD
Freeze Video	<code>#awDVDFreeze</code> <code>#awDVDFreezeNever</code> <code>#awDVDFreezeLastFrame</code>	DVD
Title Number	<code>#awDVDTitle</code>	DVD
User Control	<code>#awDVDUserControl</code>	DVD
Display Frame Numbers	<code>#awDVDFullScreen</code>	DVD
Display Video	<code>#awDVDVideo</code>	DVD
Audio Channel 2	<code>#awDVDCaptions</code>	DVD
Audio Channel 1	<code>#awDVDAudio</code>	DVD
DVD Start Time	<code>#awDVDStartTime</code>	DVD
DVD End Time	<code>#awDVDEndTime</code>	DVD
Stop Video	<code>#awDVDStopIf</code>	DVD
Stop on Keypress	<code>#awDVDStopKeypress</code>	DVD
X-Base	<code>#awDVDBaseX</code>	DVD
Y-Base	<code>#awDVDBaseY</code>	DVD
X-Initial	<code>#awDVDSizeX</code>	DVD
Y-Initial	<code>#awDVDSizeY</code>	DVD

New variables

New variables are available for the DVD media feature in Authorware 7.

For complete details, see [System variables in Authorware Help](#).

Variable	Description
DVDCurrentTime	Contains the current time, measured in seconds, of the current DVD title.
DVDState	Represents the current playback state of the DVD.
DVDTotalTime	Contains the total time, measured in seconds, of the current DVD title.
DVDWindowHeight	Contains the height of the DVD window.
DVDWindowWidth	Contains the width of the DVD window.

CHAPTER 2

Authoring Basics

Multimedia piece is a broad term that refers to many types of communication. For example, an intranet-based training course, an interactive kiosk, an online magazine, and a computer simulation of industrial equipment are all multimedia pieces. Their common multimedia trait is the ability to orchestrate media objects—such as graphics, sound, digital movies, or a set of instructions—in response to changing conditions such as the user's choices and the amount of time that passes.

The assembling of multimedia into a functioning piece is known as authoring.

Icon-based authoring

The best introduction to icon-based authoring is the Macromedia Authorware tutorial. See the online tutorial at www.macromedia.com/support/authorware, or select Help > Support Center. If you've skipped it, the information in the first few pages of this chapter will get you started on your own. Be sure to see [“Authoring—step-by-step procedures” on page 25](#) for the topics in Authorware 7 Help that explain how to use icons.

In Authorware, you construct, or author, a multimedia piece by assembling icons on a flowline. The flowline organizes the icons and determines the sequence in which Authorware runs them.

Icons contain the contents of a piece. Different types of icons contain different types of objects, such as graphics, text, sound, digital movies, or a set of instructions. The arrangement of icons forms the logic of a piece—its structure or architecture. The logic of a piece gives the piece shape and makes it work a certain way.

In Authorware, you drag icons from the Icon palette. The Icon palette is like a bottomless well. You can drag up to 32,761 icons from the Icon palette into a single file, but it is best to create smaller modular pieces, which are much easier to work with.

The flowline appears in the Design window. That's where you arrange the icons and construct the logic of the piece. The icons' order on the flowline determines the order in which events take place when the piece runs.

When you run a piece, it appears in the Presentation window. As you build a piece, you use the Presentation window to lay out text, graphics, buttons, and all the other visual elements that make up the piece.

Running the piece

As the author, you can run the piece to see how images appear and whether interactivity works the way you want. Use the Control menu and the Authorware Control Panel to run or step through your piece. Use the start and stop flags to test a specific segment of the piece.

The stages of authoring

Authoring a multimedia piece generally follows the traditional development process of analysis, design, development, evaluation, and distribution.

Analysis

Before you start using Authorware, conduct an analysis to determine what the piece you're creating needs to do. The analysis you make will differ depending on the type of solution that you are providing. Who is the intended audience? What objectives must be met? How timely must the information be? What environment will users be in when they use the piece? If you're developing a kiosk that will be situated in a noisy shopping mall, you may decide against using sounds in your piece.

Next, think about how you'll distribute the piece and the kinds of computers that the target audience has. For more things to consider before you begin designing your piece, see [“Distributing your piece” on page 50](#).

Design

Before you develop the navigational structure of your piece, make a prototype of the screen design and identify all of the content elements for your piece.

Create storyboards and prototypes Create storyboards for the project and design a few sample screens to work out in detail exactly what you want to achieve. Authorware is ideal for creating prototypes. You can draw basic shapes to represent elements on the screen and later import finished graphics into the piece. A prototype should convey the idea of how a piece will work. Be careful not to spend too much time on visual details for the prototype: If the design needs to change later, you will have created graphics that won't be used.

Identify and gather content When professional developers identify and gather their content, they create what's known as an asset list—a list of every sound, graphic, movie, and object used in a piece. You should do the same. Authorware excels at integrating text, graphics, sound, and digital movies. Although you can create some text and graphics with Authorware tools, you'll get the best results with applications designed specifically for creating the type of media you want to include. For example, use Macromedia Flash for digital movies and animations.

Use the Authorware media libraries and external content features as much as possible. They let you store content externally and reuse and reference the icons, saving disk space and making it easier to update applications.

Be sure that your computer and all end users' computers have the equipment needed to play sound and movies.

Development

Now you're ready to build full functionality into the piece, which is best done module by module. You'll need to import content, apply transition and motion effects, create navigation structures and interactions, and develop data variables.

Look for ways to set up logic so you can save it as a model and reuse it. Remember that you can switch from Authorware to external applications—for example, to let the user enter numbers in a real spreadsheet or save results in a text file. Run and test the piece regularly as you build it.

Evaluation

When you think your application works the way you want it to, test it just as software companies test applications—on real users.

Testing and fine-tuning Run the files and make adjustments to the content and the interactions. You might need to modify the files to obtain the best performance on slower computers. It's important to allow ample time for this stage; designing interactions is considerably more complicated than creating a linear presentation.

Review In this stage of testing, you make sure the project works as designed. You might need to go back to the development stage, or you might simply need to fine-tune some details.

Distribution

When you're ready to distribute your piece, package it. You can use one-button publishing to package it in multiple formats at the same time. If you're developing commercial applications, make sure your distribution is in accordance with the license agreement. The packaged file doesn't contain all the information needed to change the application, so be sure to archive the original Authorware file.

If you're developing a piece for the Internet or for a corporate intranet, you can use one-button publishing to create the HTML page where you want to embed the piece, and then put all the files on your web server. Run at least one test with each browser that will be used to view the piece, such as Netscape Navigator or Microsoft Internet Explorer, to make sure the piece downloads and runs properly.

Authoring—step-by-step procedures

The steps involved in authoring a multimedia piece are presented in the following table.

What you can do	In Authorware Help, see
Create a new file	Creating a new project file
Use a Knowledge Object in a file	Selecting a Knowledge Object for a new file, Adding a Knowledge Object to a file
Open a file	Opening a file
Build a piece	Building a piece
Set up icon properties	Modifying icon properties
Edit an icon	Editing an icon
Add text	Working with text

What you can do	In Authorware Help, see
Add images	Importing a graphic, Pasting a graphic, Drawing graphics
Display the content of several icons in the Presentation window at the same time	Displaying the content of several icons
Create effects when you display or erase images	Choosing a transition
Move objects across the Presentation window	Using the motion icon
Add sound	Working with sound, Linking to external files
Add digital movies	Working with digital movies, Linking to external files
Play DVDs	Working with DVDs
Group icons	Grouping and moving icons
Move icons from one part of the flowline to another	Grouping and moving icons
Reuse content	Using libraries
Create interactivity	Ways to interact
Track a user's performance	Tracking performance, Using Knowledge Track to set up a course
Track information	Using a variable, Creating a variable
Set up paging and navigation structures	Ways of directing the flow, Paging structures—quick and easy user control
Add functions to a piece	Using a system function
Extend functionality of Authorware	Using scripting Xtra functions, Loading a DLL custom function, Using an ActiveX control
Reuse flowline structures	Using a model
Test a piece	Running and pausing a piece
Use the Authorware Control Panel	Using the Control Panel
Save a file	Saving a file
Package a piece for distribution	Packaging an Authorware piece, Using one-button publishing to package a piece
Process a piece to play it over the web	Getting a piece ready for the web

The toolbar

The Authorware toolbar provides quick access to some of the most frequently used menu commands. Use the View menu to show or hide the toolbar.

To find out what a button does, position the pointer over it. A description of the button appears after a few seconds.

The Icon palette

The Icon palette contains 14 icons you can use to create Authorware pieces. To create your pieces, drag icons from the Icon palette onto the flowline to produce a sequence of actions that Authorware carries out. The Icon palette is customizable, dockable, and movable.

For a description of each icon in the Icon palette, see [“Icon basics” on page 28](#).

The toolbox

The toolbox appears when you open a display or interaction icon for editing. Use the tools in the Authorware toolbox to select, create, modify, and set attributes for text and graphics in the Presentation window.

For a description of each tool in the toolbox, see [“Creating and integrating images and graphics” on page 88](#).

The Tools panel

The Authorware Tools panel provides easy access to Knowledge Objects, functions, and variables. By default, the Authorware Tools panel is docked in groups in the docking area on the right side of the workspace. You can undock panel groups, undock individual panels, rearrange the order of docked panel groups, and collapse and close panel groups. You can also open and close individual panels.












The Property inspector




The Property inspector displays—and allows you to modify—the properties of selected icons. Each icon has attributes and options associated with it, which together are known as the icon’s properties. The options available in the Property inspector depend on the type of icon you have selected.

By default, the Property inspector is docked at the bottom of the workspace.

Icon basics

The following table describes what each icon does:

Icon	Description
Display icon 	Displays text and graphics on the screen. See “Working with text” on page 61 and “Working with images and graphic objects” on page 87 .
Motion icon 	Moves an object along a path or to a specific point. You can restrict where an object can move. See “Using the motion icon” on page 154 .
Erase icon 	Erases any or all objects from the screen when they are no longer needed. See “Setting up an erase icon” on page 30 .
Wait icon 	Adds pauses with or without onscreen buttons that let users decide when to continue. See “Setting up a wait icon” on page 31 .
Navigate icon 	Sets up automatic or user-controlled navigation in a hypermedia framework. See “Setting up a navigate icon” on page 31 .
Framework icon 	Creates a hypermedia framework, a structure of text, graphics, sound, animation, and data that users can navigate through. See “Setting up a framework icon” on page 31 .
Decision icon 	Sets up various paths Authorware can take depending on conditions and events. See “Setting up a decision icon” on page 32 and “Decision structures—directing the flow automatically” on page 133 .
Interaction icon 	Gives users a variety of ways to make choices, such as clicking buttons or hot spots. Evaluates a user’s actions. See “Setting up an interaction icon” on page 32 .
Calculation icon 	Updates values in variables. Executes functions that affect what the piece does. See “Setting up a calculation icon” on page 32 and “Where you can use variables, functions, expressions, code snippets, and scripts” on page 158 .
Map icon 	Simplifies and organizes the flowline by grouping it into smaller segments. See “Setting up a map icon” on page 33 .
Digital movie icon 	Plays Director 5 and earlier, AVI, MOV, FLC, MPEG, and QuickTime 2 digital movies. You will need to use the QuickTime Xtra to import QuickTime 3 or later movies. See “Setting up a digital movie icon” on page 33 and “Working with digital movies” on page 95 .
Sound icon 	Includes music, narration, or sound effects in a piece. See “Setting up a sound icon” on page 34 and “Working with sound” on page 92 .
DVD icon 	Includes still images, sound, or movie from DVD. See “Setting up a DVD icon” on page 34 and “Working with DVDs” on page 99 .

Icon	Description
Knowledge Object icon 	Includes Knowledge Objects. See “Using Knowledge Objects” on page 37 .
Start and stop flags 	Speeds up development by letting you test segments of the flowline as you create them. (The start and stop flags are usually used together.) See Running and pausing a piece in Authorware Help.
Icon color palette 	Organizes icons by color-coding them. See Coloring Icons in Authorware Help.

Some icons share common traits:

- Display, digital movie, and DVD icons display objects in the Presentation window. Interaction icons can display objects along with objects in other icons.
- Motion and erase icons affect icons that display content.
- Digital movie and sound icons can be synchronized with the contents of display icons. For more information, see Media synchronization in Authorware Help.
- Framework, decision, interaction, and calculation icons control the piece’s branching and navigation.
- Map icons contain groups of other icons. Each map can be opened to display its segment of the flowline in its own Design window.
- Sprite icons add special functionality to Authorware through sprite Xtra extensions. Sprite icons come from the Insert menu. See Using a sprite Xtra in Authorware Help.

Inserting an icon

Drag an icon from the Icon palette to the place on the flowline where you want to insert it. Authorware adds the icon to the flowline where you release the mouse button.

Setting up icons

See the topic for the specific type of icon you want to set up:

- [“Setting up a display icon” on page 30](#)
- [“Setting up a motion icon” on page 30](#)
- [“Setting up an erase icon” on page 30](#)
- [“Setting up a wait icon” on page 31](#)
- [“Setting up a navigate icon” on page 31](#)
- [“Setting up a framework icon” on page 31](#)
- [“Setting up a decision icon” on page 32](#)
- [“Setting up an interaction icon” on page 32](#)
- [“Setting up a calculation icon” on page 32](#)
- [“Setting up a map icon” on page 33](#)
- [“Setting up a digital movie icon” on page 33](#)

- “Setting up a sound icon” on page 34
- “Setting up a DVD icon” on page 34
- “Setting up a sprite icon” on page 34
- “Using Knowledge Objects” on page 37

Setting up a display icon

Use the display icon to display text and graphics. Drag a display icon onto the flowline, and then double-click the icon to add graphics and text to it in the Presentation window.

To add contents to a display icon’s Presentation window, use the toolbox to draw objects, use the File > Import command to import images, or drag and drop text or graphics files into the display icon.

To modify the display and layout of a display icon, select the icon in the flowline, and then select Modify > Icon > Properties. The Property inspector appears at the bottom of the screen and allows you to modify icon properties.

To keep previously displayed items in the Presentation window when you open a display icon, hold down the Shift key while you double-click the icon.

Setting up a motion icon

Use motion icons to move display objects from one location on the screen to another.

To move display objects:

- 1 Place a display icon on the flowline.
- 2 Place a motion icon after it.
- 3 Define the motion of the display objects in the Property inspector.

All the display objects in that icon move together. You can also move movies playing on the screen this way.

Also in the Property inspector, you can set type of motion, timing, and other options. You can preview the type of motion you’ve set up by clicking the Preview button.

Setting up an erase icon

Use the erase icon to erase any icon that’s currently displayed. You can erase the objects displayed by the display, interaction, framework, and movie icons.

When you erase an icon, you erase all its contents. For example, if a display icon contains three graphic objects, Authorware erases all three objects. If you want to erase just one of those objects, place it in a separate display icon, so it appears as a separate object in the Presentation window.

You can modify erase icon properties in the Property inspector.

Setting up a wait icon

Use the wait icon to pause the piece for a specified amount of time or until the user presses a key or clicks the mouse.

You can modify wait icon properties, such as events and time limit, in the Property inspector.

When synchronizing sounds within a piece, don't use the wait icon. For more information, see [“Synchronizing audio, DVDs, and graphics” on page 93](#).

Setting up a navigate icon

Icons attached to framework icons are called pages. You can use navigate icons to create links that jump to specific pages in a framework. When Authorware encounters a navigate icon, it goes to the page you set up as a destination. This is called a navigation link.

When you define a navigation link, you define the type of destination, such as going forward or backward a page, jumping to a page, searching for text, or jumping to the first or last page of a framework. You can select destination options in the Property inspector.

You can use navigate icons in two ways:

- Automatic navigation places a navigate icon anywhere on the flowline. When Authorware encounters the navigate icon, it automatically executes the link and goes to the destination page.
- Navigation controlled by the user attaches one or more navigate icons to an interaction icon to create a navigation framework. The interaction icon can be on the main flowline or in the entry pane of a framework icon. For each navigate icon you attach to an interaction icon, you can set up a navigation control (such as a button) that takes the user to a destination page.

For more information about navigation frameworks, see [“Frameworks in depth” on page 139](#).

You can also use navigation styles to set up hyperlinks by creating hot text. When the user clicks the hot text, Authorware executes the hyperlink and goes to a destination. For more information, see [Setting up a navigation link in Authorware Help](#).

Setting up a framework icon

The framework icon provides an easy way to set up navigation in an Authorware piece. A set of navigation controls is built into the framework icon so that users can move among the icons you've attached to it.

An icon attached to a framework icon is called a page. A page isn't limited to a display icon containing text and graphics. It can also be a digital movie, a sound, or an animation. In fact, if you use a map icon, you can make the page as complex as you want. It can contain an interaction, a decision, another framework icon with its own navigation structure, or any combination of icons you want to arrange on the flowline.

Use the Framework window to set up icons that Authorware executes when entering or exiting a navigation framework.

You can select page transition options in the Property inspector.

Setting up a decision icon

You set up a decision structure by attaching icons to a decision icon. When Authorware encounters a decision structure, it branches to a path according to the criteria you've defined.

When you set up a decision structure, you can define the following settings in the Property inspector:

- The Repeat setting determines how many times Authorware loops through the decision icon.
- The Branch setting determines to which icon Authorware branches.

Setting up an interaction icon

You use interaction icons to set up interaction structures that consist of an interaction icon with result icons attached to it. Each result icon corresponds to a response that the user can make, such as clicking a button or entering text, or to an event, such as the number of tries the user made.

When Authorware encounters an interaction icon, it displays any text and graphic elements it finds, including buttons, hot spots, text entry fields, and so on, defined by the response type symbols. Authorware then halts and waits for a response from the user. When a user responds, Authorware sends the response along the interaction flowline to see if it matches any of the target responses. If a match is found, Authorware branches to or executes the specified result icon.

You set up the responses you're looking for in the Response Type dialog box. Here you determine the user's response or the event by defining each result icon as a particular response type. When the user makes a response or the event occurs, Authorware branches to the corresponding result icon and displays its contents.

You set the characteristics of the interaction display using the Property inspector. Here you can specify display options, such as layer level, transition, positioning, and when objects are erased.

Double-click an interaction icon on the flowline to open the Presentation window to edit the display contents.

Setting up a calculation icon

Use calculation icons to execute expressions and scripts that change values in variables or that perform calculations in a piece. Insert calculation icons at any point on the flowline or attach them to other icons.

To create an expression for a calculation icon, double-click the icon to open the Calculation window. In addition to using Authorware Script Language to enter variables, functions, expressions, code snippets, and scripts in the window, you have the option of entering JavaScript code. See Chapter 9, [“Using Variables, Functions, Xtra Extensions, and More” on page 157](#). For information about using JavaScript in the Calculation window, see [“Using JavaScript in Authorware” on page 251](#).

Authorware usually executes all the statements in a calculation icon and then exits the icon. You can force Authorware to exit calculation icons by entering an Exit statement in an expression. If the calculation icon is attached to another icon, that icon is executed normally.

In most cases, it's a matter of preference whether to place a calculation icon before another icon or whether to attach it. However, if you attach a calculation to a decision or interaction icon, Authorware executes the calculation repeatedly if the icon uses Try Again branching. If the calculation icon is placed before a decision or interaction icon, Authorware executes it only once.

While Authorware executes a calculation icon, no other processing takes place and Authorware doesn't respond to mouse clicks or key presses. For calculation icons that contain relatively simple calculations, this delay is imperceptible. However, calculation icons that contain complex repeat loops take longer to execute and can cause a noticeable delay.

You can set additional properties for the calculation icon in the Property inspector.

Setting up a map icon

You can use map icons to group a sequence of icons. By grouping icons in map icons, you can organize your piece into convenient modules. This gives you a clear overview of how the piece will work, no matter how many icons it contains. Place map icons anywhere on the flowline and attach them to interaction icons, decision icons, or framework icons. You can place map icons within other map icons.

Each map icon has its own flowline. Double-click a map icon to open it. A level caption appears in the upper right corner.

When Authorware encounters a map icon, it executes the map's icons in the order in which they appear on the flowline. When it finishes executing the last icon, Authorware exits the map icon and executes the next icon on the main flowline.

If the map icon is attached to an interaction icon, Authorware executes the entire contents of the map icon before branching to the next result icon. Authorware also executes the entire contents of the map icon when it is a path attached to a decision icon or pages attached to a framework icon.

To group icons into a map icon, the icons must be next to one another on the flowline. To create a map icon, select the icons, select **Modify > Group**, and name the resulting map icon. To redisplay (or ungroup) the icons in a map icon, select the map icon and select **Modify > Ungroup**.

If you attach a calculation icon to a map icon and then ungroup the map icon later, Authorware attaches the calculation icon to the first icon that was within the group.

You can set additional properties for the calculation icon in the Property inspector.

Setting up a digital movie icon

You can use digital movie icons to import digital movies created by other applications. (To import QuickTime 3 or later movies, use the QuickTime Xtra.)

Use the Property inspector to control the movie's properties, such as its positioning, the number of times it is played, and the speed at which it is played.

A digital movie can be stored internally or externally, depending on the type of movie.

- Internally stored movies are stored in the Authorware file, increasing the size of the file. You can use erase transitions with internally stored movies.
- Externally stored movies are not stored in the Authorware file. You can't use erase transitions with externally stored QuickTime, MPEG, or AVI movies. To use an erase transition on other types of external movies, select the erase icon and the Prevent Crossfade option in the Property inspector.

You must keep externally stored movies where Authorware can open them when it encounters movie icons on the flowline. You must distribute the files for externally stored movies along with your packaged Authorware piece.

Authorware and the Authorware runtime application (Runa7w32.exe) look for external movies in the directories set in the Search Path field of the File Properties dialog box, or in the SearchPath variable, or in the external filename stored in the Movie Icons Properties dialog box.

To preview a movie, select the digital movie icon, and use the playback controls in the Property inspector to run the movie.

Note: Because movie icons are branching icons, they cannot be direct children of interaction, decision, or framework icons. You can, however, place a map icon next to the interaction, decision, or framework icon, and then place the movie icon inside the map icon. If you try to attach a movie icon to an interaction, decision, or framework icon, it will automatically be placed in a new map icon.

Setting up a sound icon

You use sound icons to play sounds in your Authorware pieces. You can import sounds and store them internally in a file, or store them externally and link them to your piece. Like other files, you can also drag and drop sound files onto the flowline of your piece.

You can play a loaded sound without opening the sound icon. To do this, click the sound icon with the right mouse button or press Control+Alt as you click the sound icon. Authorware plays the sound.

You can modify the sound properties in the Property inspector.

Note: Because sound icons are branching icons, they cannot be direct children of interaction, decision, or framework icons. You can, however, place a map icon next to the interaction, decision, or framework icon, and then place the sound icon inside the map icon. If you try to attach a sound icon to an interaction, decision, or framework icon, it will automatically be placed in a new map icon.

Setting up a DVD icon

Use the DVD icon to include a DVD movie in an Authorware piece. The DVD icon requires that your computer has a DVD drive, including the necessary codecs and drivers, and Microsoft DirectX version 8.1 or higher installed. To include a DVD movie, simply place a DVD icon on the Authorware flowline where you want the movie to play, and set its properties in the Property inspector. In the Property inspector, you can set the video, timing, and layout properties of the DVD movie.

Setting up a sprite icon

A sprite is a functional element such as a 3D object, a Macromedia Flash animation, an HTML browser window, or a pop-up menu. Sprite Xtra extensions make it possible to incorporate such sprites into an Authorware piece. When you place a sprite Xtra file in the Xtras folder and then start Authorware, the Xtra file adds a command to the Insert menu. For example, the QuickDraw 3D Xtra adds the command ThreeD to the Insert menu.

Each sprite Xtra has both a menu category and a name. The menu category appears on the Insert menu, and the name appears on the submenu. In the case of the QuickDraw 3D Xtra, its menu category is ThreeD, and its name is QuickDraw 3D Model.

To incorporate a sprite into an Authorware piece, position the Paste Hand on the flowline where you want the sprite to appear, and then select the Xtra you want from its submenu. Authorware displays a dialog box that lets you select the sprite you want, and then inserts it on the flowline in a sprite icon. When you run the piece, the sprite appears in the Presentation window.

Tips for working with icons

Here are tips for working with icons more efficiently.

Naming icons Give each icon in the flowline a meaningful name. If you're working in a team, this makes the logic much clearer for others looking at the piece. As a piece grows in complexity, or when you need to modify the flowline, icon names that describe the modules or the actual icons are invaluable.

Selecting more than one icon You can drag a selection rectangle around several icons on the flowline to select them. The Cut or Copy command will then apply to all the selected icons. You can also apply the Group command to several selected icons to place them all in a single map icon.



Rearranging icons You can rearrange icons by dragging them. When cutting and pasting icons, Authorware inserts them at the Paste Hand.

Embedding and linking content You can embed or link content directly to the flowline by using the File > Import command. Authorware looks at the file type and automatically places the correct type of icon (sound, display, or digital movie) at the Paste Hand. The Link to File option is available from the Import dialog box. Linking to a file stores the file separately from the Authorware file. For more information on linking files, see [Chapter 4, “Working with Content,” on page 59](#).

Using the drag-and-drop feature You can use the drag-and-drop feature to quickly import content into your piece. For example, you can drag files (movies, sound, images, and text) from the Windows Explorer onto the flowline. You can also drag images and text directly onto a display icon. Press the Shift key as you drag to link content, rather than embed it.

You can drag several files onto an interaction icon, making each file the result of a response. Or drag several files to a framework icon so each becomes a page. If you drag files to a decision icon, each becomes a path. If you drag files to the flowline, they're arranged one after another. Authorware automatically inserts the appropriate icon for each type of file in the flowline.

Using the Property inspector to edit icon properties Each icon has attributes and options associated with it, which together are known as the icon's *properties*. You can modify icon properties using the Property inspector. The options available in the Property inspector depend on the type of icon. For example, the display icon's properties include name, size, layout, transition, and layer. See *Modifying icon properties in Authorware Help*.

Image properties The display icon can contain more than one image. Each imported or linked image has its own Properties dialog box with options for transparency, foreground and background colors, positioning, scaling, and cropping. Select an image and then select Modify > Image Properties to view the Properties dialog box for the selected image.

Grouping icons You can group several related icons on a flowline into a map. This helps organize your information and makes it easier to find. See *Grouping and moving icons in Authorware Help*.

Coloring icons You can use a color-coding scheme for icons on the flowline or in the library simply by selecting the icon and then selecting a color from the Color palette. See *Coloring icons in Authorware Help*.

Media synchronization You can place icons to the right of sound or movie icons, in much the same way you set up decision icons. The icons placed to the right of another icon are called *child* icons. In addition, you can set those child icons to trigger based on the media position or number of seconds from when the media starts playing. See Media synchronization in Authorware Help.

Customizing icons If the icons on the Icon palette are not set up with the options you most commonly use, you can create your own customized icons and drag them back to the Icon palette. These icons become the default when you select an icon from the Icon palette. See Customizing icons in Authorware Help.

Editing an icon

You can edit an icon by double-clicking it.

When you open and edit icons that display their contents on the screen (such as the display, digital movie, or DVD icons), the Presentation window opens as well.

The toolbox appears when a display or interaction icon is open for editing in the Presentation window. The tools that are available depend on the type of icon that is open.

For display, digital movie, sound, and DVD icons, and displays within interaction icons, the Modify menu provides more options. The menu items available are the ones that can be applied to the type of icon that is currently active.

For motion and erase icons, the Presentation window also shows the contents of their associated icon or icons.

The Presentation window doesn't appear when icons that don't use it, such as navigate, wait, and calculation icons, are opened for editing. Instead, Authorware opens the corresponding options dialog box or window.

Editing multiple icons

Use the Property inspector to simultaneously change one or more properties common to the icons you've selected.

To edit multiple icons:

- 1 Select the group of icons you want to modify.
- 2 In the Property inspector, select a category, then a property, and then the setting you want. Repeat the process to change as many settings as you want.
- 3 Click OK.

To apply the same settings to a different selection of icons, select Edit > Change Properties Again.

Displaying the content of several icons

To display the content of more than one icon:

- 1 Double-click the first icon to open it.
- 2 Shift-double-click subsequent icons.

To open a set of neighboring icons:

- 1 Run the segment that contains the icons.
- 2 Shift-double-click the icons that you want to edit.

Displaying the Design window

To display the Design window containing the current icon, do one of the following:

- Select View > Current Icon.
- Press 0 on numeric keypad.

To display a Design window by name:

- Select Window > Design > Design window name.

The main Design window has the same name as the file. Other Design windows have the same name as the map icon they correspond to.

Using Knowledge Objects

Knowledge Objects can help increase your productivity with Authorware 7. Each Knowledge Object has an icon and a wizard. You incorporate a Knowledge Object into your own project by dragging its icon from the Knowledge Objects window. Some Knowledge Objects are automatically added to your file by other Knowledge Objects or by the New File dialog box. You can edit each Knowledge Object's properties with its wizard.

To use Knowledge Objects, you should have a basic understanding of Authorware icons and the flowline. Knowledge Objects developed by Macromedia have introduction screens that describe the Knowledge Object and online help screens that provide detailed instructions for specific tasks.

It's important to save your work before inserting Knowledge Object icons and launching Knowledge Objects. This is because many Knowledge Objects, especially those that use external media and U32s, rely on pathnames relative to the current file's location. Additionally, saving your work creates backup versions to which you can revert if necessary.

Selecting a Knowledge Object for a new file

To select a Knowledge Object for a new file:

- 1 Select File > New > File.
- 2 Select a Knowledge Object in the New File dialog box, and click OK.
The Knowledge Object is added to your file automatically.
- 3 Set the options in the wizard. When you're finished, click Done.

Adding a Knowledge Object to a file

To add a Knowledge Object to a file:

- 1 Select File > Save to save your file.
- 2 If the Knowledge Object window isn't open, select Window > Knowledge Objects.
- 3 Drag a Knowledge Object icon from the Knowledge Objects window to the flowline.
Knowledge Objects are grouped by category. Use the Category pop-up menu in the Knowledge Objects window to view a specific group. Select All to view all of the available Knowledge Objects. For information on adding Knowledge Objects to the Knowledge Objects window, see [“Displaying Knowledge Objects in the Knowledge Objects window” on page 214](#).
- 4 Double-click the Knowledge Object icon to launch the wizard.
- 5 Set the options in the wizard. When you're finished, click Done.

If you need to make changes, double-click the Knowledge Object icon to run the wizard again.

Authorware Knowledge Objects

Authorware 7 includes many Knowledge Objects that you can use right away. Knowledge Objects developed by Macromedia are organized into the following categories:

- [“Assessment Knowledge Objects” on page 39](#)
- [“File Knowledge Objects” on page 40](#)
- [“Interface Components Knowledge Objects” on page 40](#)
- [“Internet Knowledge Objects” on page 41](#)
- [“New File Knowledge Objects” on page 38](#)
- [“RTF Objects Knowledge Objects” on page 41](#)

Knowledge Objects created by independent developers are also available for use with Authorware 7. For more information, see [Chapter 11, “Creating Knowledge Objects,” on page 213](#).

New File Knowledge Objects

The Application and Quiz Knowledge Objects are available when you first start Authorware to create a new file. Use them as the starting point for training, presentations, testing, courseware, and other custom applications.

New File Knowledge Objects are intended for prototyping and demonstrating capabilities for new applications and quizzes. The appearance and behavior of these have been predefined. To create production applications and quizzes that meet all of your particular requirements, you should instead build your applications and quizzes using the complete range of Authorware icons and facilities.

Application Knowledge Object

The Application Knowledge Object provides navigation for sections, pages, and quizzes. Select from a variety of layout, delivery, user interface, and other options. You can customize this template with your own content for a particular discipline or instructional strategy.

Quiz Knowledge Object

The Quiz Knowledge Object is a basic testing shell that sets global parameters for the following types of exercises:

Multiple Choice is a standard question type with multiple distractors. The user can select all the answers that apply to the question.

Single Choice is similar to the Multiple Choice exercise. The user selects one possible answer for the question.

True/False is a subset of the Single Choice exercise type with only two distractors.

Short Text is a standard fill-in-the-blank question type.

Drag and Drop is a standard exercise type for matching and ordering exercises.

Hot Object is an exercise type for identification and exploration exercises that require users to click on an object in the Presentation window.

Hot Spot is an exercise type for identification and exploration exercises that require users to click on a particular area in an image.

The exercises are available with the stand-alone Quiz Knowledge Object or as part of a quiz in the Application Knowledge Object. You can also add each component separately by selecting an Assessment Knowledge Object (see [“Assessment Knowledge Objects” on page 39](#)).

Assessment Knowledge Objects

These Knowledge Objects work in conjunction with the Assessment Login and Scoring Knowledge Objects.

Drag and Drop Question is a standard exercise type for matching and ordering exercises.

Hot Object Question is an exercise type for identification and exploration exercises that require users to click on an object in the Presentation window.

Hot Spot Question is an exercise type for identification and exploration exercises that require users to click on a particular area in an image.

Login implements the login sequence and storage type to facilitate score recording for Assessment Questions in Assessment Scoring.

Multiple Choice Question is a standard question type with multiple distractors. The user can select all the answers that apply to the question.

Scoring implements score accumulation, display and recording for Assessment Questions.

Single Choice Question is similar to the Multiple Choice exercise. The user selects one possible answer for the question.

Short Answer Question is a standard fill-in-the-blank question type.

True/False Question is a subset of the Single Choice exercise type with only two distractors.

Note: The Hot Spot, Hot Object, True/False, Single Choice, and Multiple Choice Assessment Knowledge Objects allow for the import and export of question content stored in XML files that conform to the IMS Question Test Interoperability (QTI) version 1.1 standard. This allows you to export a question to an XML file and import an XML file. (See [“QTI Assessment Knowledge Objects” on page 185](#).)

File Knowledge Objects

Add/Remove Font Resource loads a font file into memory so that Authorware can draw text using the font.

Copy File copies one or more files into a directory.

Find CD Drive locates the CD-ROM drive on the computer running your piece.

Jump to Authorware File jumps to other Authorware files from the current piece.

Read INI Value reads a value from a Windows INI file.

Set File Attribute sets the system attributes (such as read-only) of external files.

Write INI Value writes a value to a Windows INI file.

Interface Components Knowledge Objects

The following Knowledge Objects let users interact with your piece more effectively. Use them as components of your own interface design.

Browse Folder Dialog Box displays a Windows-style dialog box for browsing and selecting directories.

Checkboxes displays a set of check box selection controls in your piece.

Message Box displays a Windows-style message or alert dialog box.

Move Cursor changes the location of the mouse pointer.

Movie Controller displays standard play, pause, stop, rewind, and fast-forward controls for video and audio files.

Open File Dialog Box displays a dialog box for selecting a file to open.

Open File Dialog displays a dialog box for selecting a file to open. It stores the absolute path to the selected file in a variable.

Radio Buttons displays a set of radio-button selection controls in your piece.

Save File Dialog Box displays a dialog box for specifying a folder and filename.

Set Window Caption changes the text in the Presentation window title bar.

Slider displays a horizontal or vertical slider control in your piece.

Windows Control displays a common Windows Control. (See [“About Windows Control Knowledge Objects” on page 232.](#))

Windows Control Get Property retrieves a Windows Control property value. (See [“About Windows Control Knowledge Objects” on page 232.](#))

Windows Control Set Property modifies a Windows Control property value. (See [“About Windows Control Knowledge Objects” on page 232.](#))

Internet Knowledge Objects

Authorware Web Player Security sets trusting and nontrusting options for the Authorware Web Player.

Launch Default Browser launches the specified URL with the system's default web browser.

Send E-mail sends a standard e-mail message within your piece.

RTF Objects Knowledge Objects

After you have created an RTF file using the RTF Objects Editor, you can add it to the flowline and edit how it functions on the flowline using Authorware RTF Knowledge Objects. For information about how to create an RTF file using the RTF Objects Editor, see [“Using the RTF Objects Editor” on page 66](#).

Create RTF Object creates a new RTF Object ID in the Authorware Presentation window. It takes your RTF document and input and returns an RTF Object ID. Use this ID in all other RTF Knowledge Objects to identify your RTF Object. For more information, see [Creating an RTF Object Knowledge Object in Authorware Help](#).

Save RTF Object exports a previously created RTF Knowledge Object to a new RTF document or graphic after it has been preprocessed, meaning that embedded variables have been replaced and external media has been imported. For more information, see [“Exporting an RTF Object Knowledge Object” on page 82](#).

Show or Hide RTF Object makes a previously created RTF Object visible or invisible during run time. (See [“Using the Show or Hide RTF Object Knowledge Object” on page 82](#).)

Get RTF Object Text Range locates a specified text string and returns the position of that text string in an RTF Object. (See [“Using the Get RTF Object Text Range Knowledge Object” on page 82](#).)

Insert RTF Object Hot Text Interaction creates an interaction linked to an RTF file and a specified number of responses. (See [“Inserting an RTF Object Hot Text Interaction Knowledge Object” on page 82](#).)

Search RTF Object locates a specified text string and returns the position of that text string in an RTF file. (See [“Searching for text in an RTF Object Knowledge Object” on page 83](#).)

CHAPTER 3

Working with Authorware

Good authoring practice

The typical Macromedia Authorware piece is complex, with many components to keep track of. To be successful in authoring, you need to organize your work.

Set up file properties first Before you start building a piece, take a look at the File Properties dialog box (select Modify > File > Properties).

You use the File Properties dialog box to set many overall options for the piece. It's important to set file properties first, because changes you make to file setup may affect the playback and interaction of the piece. You can change the file properties later if you need to.

The File Properties dialog box includes the following options:

- Background color for the Presentation window
- Chroma key color
- The size of the Presentation window
- Whether the Presentation window has a menu bar, title bar, or task bar
- The default Wait button
- The transition when a user returns to the file
- Which folder Authorware searches for external files
- Whether to run the piece from the beginning or from where the user left off

Break up the flowline into manageable sections By placing groups of related icons in map icons, you can organize your work into logical chunks of related material. Grouping icons helps you understand how all the icons in the piece affect each other. Grouping also helps you locate the source of problems if things don't work right.

Create a naming scheme Descriptive names help both you and anyone else who's working on or troubleshooting your piece. Develop a naming scheme for all icon names, variable names, keywords, and filenames (both Authorware files and external content files). The names should be consistent and descriptive, have a clear pattern, and indicate what an icon is for, what a variable does, or what a file contains.

One naming scheme is to prefix all your variable names with a letter that corresponds to the icon in which it is used. (To improve readability, use the underscore character after the prefix.) This scheme lets you easily identify the type of icon the variable is used in and alphabetically groups the variables in the Variables window. Another naming scheme is to suffix media files with numbers that indicate the latest revisions.

Use keywords When you're searching for an icon you've created, the text it contains provides an easy way to identify it. In a multimedia piece, however, information is available in forms other than text—digitized sounds and music, voice-overs, animations, digital movies, and video recordings, to name just a few. To make any kind of icon easy to find, assign a keyword to it.

Think about rhythm and pace Most people are sophisticated viewers. They're used to films and television shows in which something is always happening. To keep users engaged, avoid dead spots when you're jumping between files or loading external media. The danger of dead spots is even greater if you're using "distant" media. Downloading the segments of a web-packaged piece over the Internet—or even reading external content from a server connected over a local area network—can be slowed noticeably by network traffic. Design your projects so that long downloads and data transfers aren't necessary. And orchestrate the flow of information so there's always something to engage a user's attention when downloads and data transfers are happening.

Comment your work The reason you assembled icons or assigned a value in the piece may not be obvious to someone who works on the piece after you. It might not even be obvious to you after a few weeks pass. Be sure to describe what important variables do and how complex interactions work. Comments on the flowline and in calculation windows save time and clarify your work. Place brief comments in icon titles and more detailed comments in calculation icons.

Save your work frequently Frequently saving your work and creating backup files will spare you the frustration of losing valuable time if something goes wrong. With Authorware you can save file revisions in one of the following ways:

- File > Save saves the revised file by replacing the older file of the same name.
- File > Save As gives the revised version a different filename and leaves the original file intact.
- File > Save And Compact gives the revised version a different filename and compacts it to minimize disk space. The compacted version returns to its uncompact size when you reopen the file.

Run the piece frequently as you author It's usually easier to identify a problem shortly after you introduce it. Studying a few steps at a time helps you focus on where the problem might be. If there's a problem in an area that ran correctly before you added and edited some icons, the new icons are almost certainly involved in the problem. If you run your piece frequently, you can easily narrow any problem down to the small number of icons you worked with since you last ran the piece. Use start and stop flags to focus on specific sections. Use the Authorware Control Panel to trace the progress of Authorware down the flowline.

Test the piece on a range of computers Run the packaged piece on all the types of computers that you expect users to have. The piece can behave very differently on different computers. For example, the speed may be slower, noticeable pauses may occur when objects are loaded into memory, or colors may be undesirable when the piece runs on a computer that uses a lower color depth. If users will be running the piece in Windows 2000, you should test the piece in Windows 2000.

Back up and archive Always back up files before you package a piece. At the end of a project, create an archive of all the project files and store it in a safe place. Two sets of archive files are best. You can use high-capacity Zip disks or recordable CDs to archive your project.

Good authoring practice—step-by-step procedures

What you can do	In Authorware Help, see
Set up file properties first	Creating a new project file
Break up the flowline into manageable sections	Grouping and moving icons
Create a naming scheme for variables	Creating a variable
Use keywords to help identify icons	Assigning a keyword to a page
Comment your work	Using comments
Save your work	Saving a file
Use start and stop flags to focus on specific sections	Running and pausing a piece
Step through a piece	Using the Control Panel
Test your piece	Using the Control Panel, Troubleshooting methods

Troubleshooting methods

Good authoring practice helps avoid problems in the first place, but no piece works exactly as planned the first time. That's a basic fact of authoring.

You'll always need to make revisions and adjustments. Although it's impossible to predict every problem that could arise, you can rely on basic troubleshooting techniques to detect and solve problems.

Authorware does exactly what you tell it to, but it doesn't always do what you expect. When something unexpected happens, it doesn't mean that the software has a bug. Most likely, something in the icons' order or the way they're set up produced the unexpected result.

The following troubleshooting methods can help you identify a problem. You'll probably want to refer to these examples as your understanding of Authorware increases.

Determine which section has the problem

Most pieces are too large and complex to analyze completely from beginning to end. Fortunately it's rarely necessary to analyze a piece as a whole; the area in which the problem occurs is usually easy to narrow down. Here are some examples of reasonable approaches to common situations:

- If clicking a button produces the wrong result, look at the interaction that contains the button.
- If a display icon doesn't appear, check its position on the flowline. It may be positioned after a wait icon that prevents Authorware from running it or before an erase icon that erases the image before you can see it.
- If a decision icon produces an unintended result, check the values that you used to control branching.

You can save time by using the start and stop flags to restrict the portion of the flowline that you run. If a problem doesn't occur when you run a section between start and stop flags, the problem is probably related to an icon outside the section you ran.

Think through the problem section

When a section of the flowline doesn't do what you want, first try to trace the flowline's behavior in your head. That often helps you find the problem.

Connections between icons aren't always apparent from looking at one section of the flowline. Use **Modify > Icon > Connections** to list all the connections to and from an icon. From the list you can go to the icon on the flowline that is at either end of the connection. If the selected icon is a framework icon or a map icon, you can also show connections between icons within that structure.

Check the icon settings

Check each icon's properties to make sure they're set correctly. For example, a display icon might appear in a layer that hides it from view. You can remove the icon from the flowline and then run the section to see if the problem persists. If the problem goes away when the icon is removed, the problem may be in the icon's settings.

Use the Property Inspector to modify the properties of selected icons. Each icon has attributes and options associated with it, which together are known as the icon's properties. The options available in the Property inspector depend on the type of icon. For example, the display icon's properties include name, size, layout, transition, and layer.

Figure out where the flow goes

Problems often occur because Authorware follows the flowline differently than you expected. For example, a response type has different results for continue and try-again branching.

The Authorware Control Panel window displays a list of icons in the order that Authorware encounters them.

Check variables and functions

Analyze the values related to variables and the usage of functions to see how the values change as the piece runs. They may do the wrong thing or change at the wrong time. See **Variables, functions, and expressions** to make sure you understand how variables, functions, and expressions work.

To make sure you've set up the variables and functions correctly, see **System variables and System functions** in Authorware Help. If you're using them correctly but the problem persists, track the values related to the variable. There are two ways to track what happens to variables as the piece runs:

- Display the variable on the screen, so you can see its value as Authorware proceeds down the flowline.
- Use the **Trace** function to display the variable's value in the Authorware Control Panel window. For more information, see the entry for the **Trace** function in **System functions** in Authorware Help.

Make changes one at a time

Don't be afraid to change things in the piece to see if a change eliminates the problem or gives a result that points to the problem. At the same time, don't trade one problem for another. Change elements one at a time and change them back if the problem isn't fixed. If you make many changes before solving a problem, you might lose sight of the original problem and possibly introduce new problems.

Re-create the section

If you haven't found the problem, try re-creating the section from scratch in a new file. Don't copy and paste icons from the old flowline: that might just copy the problem.

By re-creating the section, you reconstruct the logic at its most basic level and verify that Authorware is working as you expect. If the section that you re-create still doesn't work properly, chances are there's something wrong in the logic for the section.

If the section that you re-create works properly, compare that section to the original piece to see where they differ. You can also copy the section into the original piece to see whether that corrects the problem.

Troubleshooting—step-by-step procedures

What you can do	In Authorware Help, see
Find an icon's connections	Icon > Connections
Check an icon's properties	Setting up icons
Check the flow	Using the Control Panel
Step through a piece	Using the Control Panel, Trace (function)
Display a variable on the screen	Displaying variable information onscreen
Check the list of common problems	Troubleshooting methods
Check the TechNotes on the Macromedia website	Help > TechNotes

Setting up file structures

File structures

In addition to the Authorware file that contains the flowline with the icons you've arranged on it, Authorware typically uses several other files when it runs a piece. These may include the following:

- Libraries linked to the piece
- Linked external files such as digital movies or sound files
- User records
- Any drivers that the piece requires
- Any custom function files that the piece requires

When running a piece, Authorware searches through specified folders in a set order until it finds these files. It is critical to make sure that files are in a folder that Authorware searches.

Default search path

When Authorware runs a piece, it must be able to find the additional files, such as the libraries and drivers that it requires. Authorware does this by searching a series of folders, so placing files in one of these folders is the simplest way to make sure that Authorware finds them.

Authorware searches directories for files in the following order:

- The directory where the file was located when you first loaded it. (When the piece is packaged or the file is moved, Authorware no longer finds the file unless you tell it the file's new location.)
- The directory where the piece is located
- The directory that contains Authorware or the Authorware runtime application (Runa7w32.exe), whichever is currently running
- The Windows directory
- The Windows System directory

Specifying file locations

You might want to put files in places other than the default folders. For example, to make the piece available to a group of people, rather than store separate copies of all the files on each local computer, you could make them available from a server. In these cases, you must tell Authorware which folder contains the file and where to find it. You can do this by specifying folders in the following places:

- In the Search Path field of the File Properties dialog box, specify folders that Authorware always searches after the piece starts.
- In the variable `SearchPath`, specify folders that Authorware searches in addition to the folders listed in the Search Path field. Because Authorware searches folders listed in the variable `SearchPath` only after the piece has started, don't include folders that you want Authorware to search for libraries. Libraries must be opened when the piece opens, so it is best to list their folders in the Search Path field in the File Properties dialog box.

When a piece is packaged, Authorware loses track of where external files were located when you loaded them. You must use one of the methods just described to make sure a packaged piece finds external files when it runs.

You can specify folders and files by their names—delimited by backward slashes(\), forward slashes(/), or colons(:).

If you specify partial pathnames that don't include the root, Authorware interprets the path as a subfolder of the folder for the current piece. However, placing a path delimiter (\, /, or :) before the partial pathname causes it to move up one level.

For example, if the folder for the current piece is C:\A7w\Myfolder and the default search path is set to Library\Media;Data\, Authorware interprets the search path as C:\A7w\Myfolder\Library\Media and C:\A7w\Myfolder\Data. However, Authorware interprets the search path \Library\Media;Data\ as C:\A7w\Library\Media and C:\A7w\Data.

It's possible that you have set up a search path correctly, but that for some reason the file or folder is missing from its proper location. Use the `FileType` function to check whether a file or folder is where you expect it to be.

See also

[“Specifying where Authorware searches for files” on page 50](#)

User records

Users often want to explore a piece part way and later return to where they left off. Authorware lets users do this by recording data about the current condition of the piece when the user quits a session. The information is saved in a separate file, called a user record.

Authorware creates user records whenever it runs a piece that has Resume selected in the On Return option of the File Properties dialog box. It retrieves the values in the user records file to determine what the location on the flowline and current custom variable values were when the user quit or Authorware jumped out of the file.

When you distribute packaged files that have Resume set in the File Properties dialog box, be sure to reserve enough space for the user records folder on the user's hard disk. The amount of space needed depends on the number of files, the number and contents of custom variables, and the number and type of objects appearing in the Presentation window when the records are saved.

Typically, you should reserve about 100K for the user records folder. Authorware automatically allocates space in the records folder for each file.

Location of user records

Authorware creates a user records file whenever Resume is selected in the On Return option of the File Properties dialog box. It retrieves the values in the user records file to determine where the user was when they quit last. The default folder for user records is `A7w_data`, which is in the user's Application Data directory. The location of this directory varies by operating system version and registry settings. The system variable `UserApplicationData` contains the complete path to this directory.

If this folder doesn't exist on the user's computer, Authorware automatically creates it when the user first runs an Authorware piece. Each user record receives a filename that consists of the first eight characters of the piece's filename and the extension `REC`.

Authorware still supports its previous behavior, for backward compatibility. For example, some developers distribute an `AWx.INI` file with their piece to control the location of the records directory and other runtime options. If the `AW7.INI` file doesn't exist in the user's Application Data directory, Authorware checks the directories it used to use, next to the application, and then the Windows directory. If Authorware finds the INI file there, it uses that location. If Authorware can't find the INI file, it defaults to writing and reading its data from the user's Application Data directory.

Changing the records folder

You can change the location of the records folder for several reasons:

- To route users to different files in a modular piece and still retain their individual records
- To set up separate records folders for individual users when multiple users work with the same piece
- To relocate records to another disk (often necessary when you are running on a network)

If user records are write-protected, Authorware is prevented from updating records as the piece runs. This commonly occurs because the user doesn't have access privileges to the folder. It's a good idea to check whether the user record is write-protected by trying to write a text file to the folder that contains user records.

You will need to use either functions or variables to change the records folder location:

- Specify the folder in the variable `RecordsLocation`.
- Specify the new records folder in the last argument of the `JumpFile` or `JumpFileReturn` function.

Specifying where Authorware searches for files

You can change the default location where Authorware searches for files.

To specify where Authorware searches for files:

- 1 Select **Modify > File > Properties**.
- 2 Click the **Interaction** tab.
- 3 In the **Search Path** field, specify the location of external files.

If you don't specify a drive in the path, Authorware defines the path relative to the location of the Authorware file.

To specify multiple search paths, separate the names with semicolons. For example, the following path searches the root level of a hard drive named `Main` and then searches the `A7W` directory of a mounted file server named `Group`:

```
Main\; Group\A7W
```

Distributing your piece

After you have completed your Authorware piece, you will need to package it in a format that can be used by the audience who will be running it.

You can distribute your work in several ways: on disk, on CD, over a network, or on the web. Authorware needs only to see an accessible disk; it makes no distinction between a local hard drive, a removable disk, or a network server.

Before you start to author your work, you need to decide which distribution medium to use—disk, CD, or network (intranet or Internet)—and where to include the runtime application. The choice will depend on the size of your files, the types of computers the piece will be played on, and the terms of your Authorware License Agreement.

The point of creating pieces is to distribute them. When you distribute an Authorware piece, you provide users with a packaged or web-packaged piece—a separate version that users can run but not edit.

Always design for the least capable system your piece will run on. If you've designed your piece with 16-bit sound, it won't be of much use to someone with an 8-bit sound card.

Here are just some of the requirements you should consider:

- Which operating systems will the piece be delivered on?
- What are the size and color capability of users' monitors?
- How much RAM and hard disk space do they have?
- Are there any other hardware requirements, such as sound cards for Windows computers or CD drives?
- Do you need to provide installation instructions, Read Me files, or an installer program?
- If you're delivering over a network, which web browsers and specific browser plug-ins, such as Macromedia Authorware Web Player and ActiveX, will your audience need?

If your piece contains libraries, you'll need to set the library search path for each distributed piece so that Authorware knows where to look for the libraries (see Packaging a library in Authorware Help).

Selecting a distribution medium

You can distribute your work in several ways: on disk, on CD, or over a local area network, a corporate intranet, or the Internet. You can create CD–web hybrids that contain an Authorware piece but receive updated information over a network.

Authorware needs only to see an accessible disk; it makes no distinction between a local hard disk, a removable disk, or a network server. The best approach depends on your particular situation and requirements.

Delivering on CD or disk

CD A single CD can store up to 740 MB of data. The popularity of CD-ROM drives and CD mastering devices makes this a good choice for many types of multimedia delivery.

Other removable disks Many other types of removable disks are available—from 3.5-inch disks to cartridges that can hold a gigabyte of information to DVDs that can hold several gigabytes of information. Make sure your users have the appropriate disk drive to read the delivery medium you choose.

If your file won't fit on one disk, you can divide the file into portions so that it can fit on several disks. In this case, include an installation program to make it convenient for users to install the piece.

Delivering across networks

Network delivery—local area, intranet, or Internet—has the advantages of being accessible and flexible. You can easily revise files, and users can run the piece at their convenience. You can track the progress and results of everyone who runs the piece.

Local area network You can distribute a piece over a local area network when your users' computers are all connected to the same network. The piece can be stored on a hard disk attached to the server or on the server itself. Users save disk space on their local computers because many of the external files, such as libraries, remain on the server.

The server doesn't have to be the same platform as the users' computers, as long as the local computers can mount it. For example, a Windows computer can obtain files from a UNIX server as long as it can mount the server.

Internet or intranet Delivery over the Internet or a company intranet has the advantages of being timely and responsive to change and of eliminating the cost of producing and distributing disks or CDs.

The main limitation of Internet or intranet delivery is the data transmission speed of users' computers and the network that carries your piece. Design your piece with the benefits of the Internet in mind—interactivity, discussion, and access to other resources—and try to keep large images and movies at a minimum. For more information on delivering your piece over corporate intranets and the Internet, see [Chapter 10, “Streaming, Intranets, and the Web,” on page 201](#).

Using a combination of disk and network delivery

CD–web hybrids offer a way of combining disk and network delivery. A CD–web hybrid contains links to files over a network. For example, you can store all the media files for a piece on a CD and have questions for the piece stored on a web server so you can easily update or change them.

Optimizing file size

Here are a few techniques for optimizing how files fit onto the distribution medium you choose:

- Break the piece into several parts so that portions of it can load at a time. It's a good idea to plan ways to make a file modular while authoring it, rather than waiting until you're ready to package it.
- Package some files to a hard drive.
- Use a compression utility such as WinZip to compress your media files, and provide an installer to expand the files.

Packaging an Authorware piece

Packaging a piece converts it to the format that you distribute. You don't need Authorware to run a packaged piece, but you do need a runtime application to play it. The runtime application can be distributed separately from the piece, or it can be part of the packaged piece.

Options for packaging a piece

Authorware provides three basic options for packaging a piece:

- One-button publishing provides a comprehensive tool for publishing and distributing a piece and all its related components in one step. This is the tool of choice in almost all publishing situations, including publishing to a CD-ROM and publishing to the web. The comprehensive settings provide almost every publishing option you need. When you initiate the one-button publishing process, Authorware automatically scans your piece and finds the files you need to publish for each medium. For more information, see [“Using one-button publishing to package a piece” on page 55](#) and Using batch publishing to package a piece in Authorware Help.
- Web-packaging a piece provides a way to publish a piece when there are special circumstances. It is used to distribute pieces over the web.
- Packaging a piece is rarely needed. It is used to distribute pieces over a LAN or CD-ROM.

The Authorware runtime application

For Windows 98SE, Windows NT, Windows 2000, Windows ME, and Windows XP, the runtime application is Runa7w32.exe. The runtime application file is in the Authorware program folder.

If you package a piece with the runtime application, the piece is a completely self-contained application. But sometimes you want to keep the runtime application separate. For example, if you have a lot of packaged files that are part of one project, all of them can use the same runtime application if it's not packaged. Also, a piece packaged with the runtime application is not cross-platform.

Do two things before you package a piece:

- 1 Make a copy of the piece. Once a piece is packaged, its structure can't be examined and its media can't be copied or edited (the flowline and icons are no longer accessible). Make sure you keep a copy of the unpackaged version of the files in case you need to make changes to the piece.
- 2 Make sure you have all the additional files that the piece requires, such as Xtra extensions and DLLs, and that you follow the license agreements related to them.

Packaging libraries

When packaging an Authorware piece that contains icons linked to a library file, you must provide the packaged library. The packaged library must be available to each computer that runs the packaged piece.

The runtime application reads libraries, but doesn't write to them. This means that libraries can be shared by more than one piece at a time over a network.

You can provide the library by packaging it within the piece or you can package it and provide it separately. Packaging and providing a piece separately avoids duplicating the library's content and saves disk space. However, you must make sure the library is available on the computer that runs the piece.

If you are using one-button publishing, you have the option of packaging your libraries internally, in which case they do not have to be distributed with the piece. However, if you do not package your libraries internally, you will need to set up publishing settings for each library, as well as the piece.

Libraries are searched for before a piece opens. The information in the content search path can't be read until the file opens, so the content search path doesn't help in finding library files (it does help with movies and such). The only way to find the library is to change the search path in the A7w.ini file, put the file in the same location as the packaged piece, or put the file in the exact same location it was in when it was packaged.

When you package libraries separately, store packaged library files in the same folder as the piece. Putting packaged libraries in this location ensures that the runtime application can find them. If the runtime application can't find a library file when opening the piece, a standard dialog box appears that asks the user to locate the library file.

When a link between an icon and the library it's linked with is broken, the icon's contents do not appear. However, Authorware still presents the icons and takes into account all other information about the icon, such as option settings and attached calculations. Even though no contents are displayed, variables can be affected.

Packaging a piece—step-by-step procedures

What you can do	In Authorware 7 Help, see
Determine which files need to be published	Determining which files you need to distribute
Package a piece	Using one-button publishing to package a piece, Using batch publishing to package a piece
Republish a piece that has been published previously	Repackaging a piece
Package external media files internally	Packaging linked media internally
Package a library	Packaging a library
Package a piece and its libraries separately	Packaging a piece and its libraries separately
Package a piece for the web	Getting a piece ready for the web, Getting ready for the web—step-by-step procedures
Add ADL-SCORM metadata to a piece	LOM Metadata Editor
Locate missing Xtra extensions	Using the Find Xtras command

Getting a piece ready for the web

After you've completed a piece you can package it for delivery over the World Wide Web. To package a piece for the web, use one-button publishing—it segments and compresses the file to make downloading faster. You can also use one-button publishing to set up how external files (libraries, Xtra extensions, digital movies, and so on) are downloaded.

You must embed the web-packaged piece in an HTML page. You must also configure a web server and put the web-packaged piece and all its external files there. This is all done with one-button publishing.

Users of a web-packaged piece must have the Authorware Web Player to run it. You can supply the player with the piece, or users can download it from the Macromedia website.

For more information on delivering an Authorware piece over the Internet or a corporate, government, or educational intranet, see [Chapter 10, "Streaming, Intranets, and the Web," on page 201](#).

Getting ready for the web—step-by-step procedures

What you can do	In Authorware Help, see
Package a piece for the web	Publishing for the Authorware Web Player, Publishing to a web page
Publish remotely to an FTP site	Publishing remotely
Republish your piece	Repackaging a piece
Combine map files for different player versions	Combining map files for different player versions
Play a Macromedia Director movie in a web-packaged piece	Playing Director movies in web-packaged pieces
Tune the performance of a web-packaged piece	Performance tips for web-packaged pieces
Configure the web server	On your server, Configuring your server
Add a web-packaged piece to a web page	Publishing to a web page
Set up a web page for non-streaming browsers	Publishing for the Authorware Web Player

Using one-button publishing to package a piece

You can streamline the publishing and packaging process by using the Publish command in the File menu. It provides a way for you to find most of the Xtra extensions, libraries, and DLLs you have used, and package and publish several versions of a single Authorware piece in one step.

For example, for a single Authorware piece, you can publish Authorware packages without runtime (A7R file), for the Authorware Web Player (AAM file), and for a web page (HTM file) by simply choosing one command.

The first time you use one-button publishing for a piece, you should look at the publishing settings to be sure you have set it up to publish your piece in the way you want to. After you have published the piece the first time, the settings you have selected remain, and you can republish it by simply using the command File > Publish > Publish. In most cases, you can produce exactly the published files you want using one-button publishing; however, you can also publish pieces without changing the default settings. You can also review each of the settings and customize them as needed. You can then save your customized publishing settings, so you do not have to reenter them if you need to package and publish your files again.

Setting up one-button publishing settings

The following are the default settings for one-button publishing:

Without Runtime publishes a packaged A7R file.

For Web Player publishes an AAM file that can be launched with the Authorware Web Player.

Web Page publishes an HTML file.

You can select from the following one-button publishing settings for CD, LAN, and Local HDD:

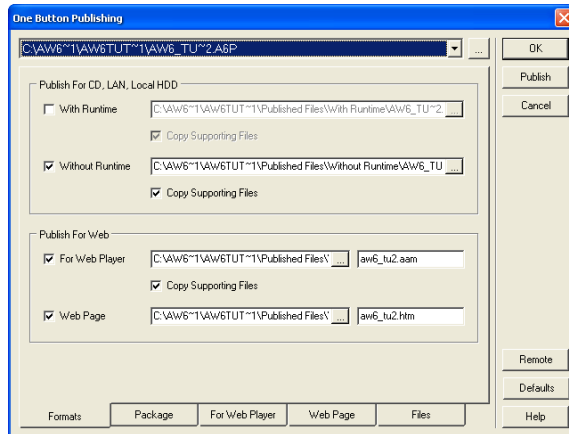
With Runtime publishes an executable EXE file. (This is not a default setting. To select it, see [“Setting up one-button publishing settings” on page 55.](#))

Without Runtime publishes a packaged A7R file (this is a default setting).

You can also select from the following settings for distribution over the web:

For Web Player publishes an AAM file that can be launched with the Authorware Web Player (this is a default setting).

Web Page publishes an HTML file (this is a default setting).



Determining which files you need to distribute

A piece typically consists of not only the Authorware file that contains the flowline, but also external files, extension functions (such as Asset Xtra extensions, scripting Xtra extensions, U32s, and DLLs). Authorware may use several other external files, depending on how the piece was developed, including GIF, JPEG, MOV, AVI, and others. To run properly, a packaged piece requires that these external files be available. If they aren't already available on the computer running the packaged piece, you must provide them.

Which files a packaged piece requires depends on the piece's components and the platform it runs on. You might need to include the following files in addition to the Authorware A7P file that contains the flowline:

- Xtra extensions for each media type used. Include the Xtra extensions that are required by Authorware to process any graphic or sound format, as well as Xtra extensions for any specific formats. For example, if you use GIF, include GIFIMP.x32. For details, see the following topics in Authorware Help:

Files you need to distribute for images

Files you need to distribute for sounds

Files you need to distribute for movies

Files you need to distribute for Director movies

Files you need to distribute with animated GIFs

- Linked files, such as linked graphics, external sounds, and Director, QuickTime 2.0 for Windows, or Video for Windows movies.
- Authorware drivers and system-level drivers for components such as QuickTime 2.0 for Windows, Video for Windows, video overlay, and videodisc players. For details, see the following topics in Authorware Help:
Files you need to distribute for movies
Other files you may need to distribute
- Fonts, when you want to make sure the piece has a specific font that might not be available on users' computers. For details, see Distributing text fonts in Authorware Help.
- Installation and decompression programs for packages that you have compressed.
- External software modules that the piece uses, such as Xtra extensions, ActiveX controls, U32s, and DLLs. For details, see the following topics in Authorware Help:
Distributing Xtras
Distributing the Macromedia Control Xtra for ActiveX
Distributing U32s and DLLs
Distributing an Authorware piece that uses a database

Commands menu

The extensible Commands menu contains several built-in commands, including the following:

Authorware Learning Object Content Packager Manager

Accessibility Kit The Authorware Application Accessibility Kit can make your learning applications accessible to all end users. Use the Accessibility Kit to tap into Knowledge Objects, Commands, models, and techniques to make text audible; run movies with captions; create accessible user interfaces, menus, and quizzes; and edit hot keys to find buttons, hot spots, and hot object responses that do not have keystrokes attached. The Accessibility Kit includes scripting to automatically detect the text-to-speech facilities already installed in the end-user's computer, including JAWS and WindowEyes.

Find Xtras This is used to assist in locating the Xtra extensions that are used for an Authorware piece and copy them all to the same folder. For more information, see [Using the Find Xtras command in Authorware Help](#).

RTF Objects Editor This is used to create and edit RTF objects without leaving Authorware, and includes enhancements not found in other RTF editors, such as the ability to add images, paste in Authorware expressions, and create your own graphics. For more information, see [“Using the RTF Objects Editor” on page 66](#).

Authorware Learning Object Metadata Editor The Department of Defense Advanced Distributed Learning (ADL) initiative has developed common guidelines for the development, delivery, and management of learning. The Shareable Courseware Object Reference Model (SCORM) is a reference model for a set of interrelated guidelines for reusability, accessibility, durability, and interoperability. This is accomplished by the use of very structured packaging guidelines, or metadata. The Learning Object Metadata Editor now provides the metadata with your package that conforms to ADL-SCORM specifications. For more information, see [“Authorware Learning Object Content Packager command” on page 186](#).

You can also create your own commands and add them to the Commands menu.

Adding your own commands to the Commands menu

When you start Authorware, it populates the Commands menu with the names of all the EXE and A7R files located in the Commands folder in the Authorware program directory.

You can add any kind of executable (EXE) file to the Commands menu. You do not have to create it using Authorware. However, only Authorware EXE, A7R, or A7P files, or files built with an SDK like the Delphi SDK (included on the Authorware CD), will be able to communicate with Authorware.

The behavior of the items in the Commands menu is similar to that of the Authorware Knowledge Objects. When you select a command, another instance of Authorware is launched that runs the command. The Commands menu uses the same `CallTarget` protocol, and the commands are modal by default. This means you cannot access the original instance of Authorware until the command is completed or stopped. With modal commands, only one command can be executed at a time. When the command completes, control is returned to the Authorware application. For instructions on how to do this, see *Adding your own commands to the Commands menu* in Authorware Help.

However, you can create nonmodal commands. This makes commands much more versatile, in that a command can act as an assistant that stays open while you make changes to your file. Using nonmodal commands, it is easier to write and use commands such as improved spell checkers, find and replace engines, bug trackers, media browsers and importers, and many other types of tools. The `SetTargetModal` function lets you switch from a modal to a nonmodal state from inside a command. For more information, see “SetTargetModal” in Authorware Help.

CHAPTER 4

Working with Content

A Macromedia Authorware piece is made up of logic and content. Logic—the way you arrange and structure icons on the flowline—defines how a piece works. Content—text, graphics, movies, and so on—communicates the information in a piece and gives the piece its look and feel.

This chapter describes how to work with content as you develop an Authorware piece, create and integrate media into a piece, and make the most of the media. It includes cross-references to instructions on using text, graphics, animation, sound, and video.

You can use several types of media in Authorware:

- Text
- Graphics (backgrounds, photos, drawings, diagrams, borders, buttons)
- Digital movies (Macromedia Director movies [Windows only], Macromedia Flash movies, animated GIFs, QuickTime 2 movies, Video for Windows movies, Windows Media Player, PICS)
- Sound (music, voice-overs, sound effects)
- DVD movies
- Other media (Flash, animated GIFs, QuickTime 3 and QuickTime 4 formats)

Creating and incorporating content

You can use Authorware to create some content (for example, text and simple graphic objects). However, most content is created outside Authorware using specific media editors. Media editors are sophisticated applications for creating and modifying digital media.

Using Xtra extensions and OLE to include content

Authorware can also be extended through the use of Xtra extensions to include other types of content, such as QuickDraw 3D objects. Xtra extensions are plug-ins that let you add functional elements to the Presentation window. For example, with the Macromedia Control Xtra for ActiveX you can embed elements such as web browsers, calendars, and user interface controls in an Authorware piece.

In Windows, you can use object linking and embedding (OLE) to embed such elements as functioning spreadsheets and charts in an Authorware piece.

Embedding and linking content

Once you've created content in a media editor, you need to import it into Authorware, either by embedding or by linking. If you embed a file, the actual file data is stored in the Authorware piece. If you link to a file, the only data stored in Authorware is the file's name and location. For information on specific procedures, see [“Embedding and linking content—step-by-step procedures” on page 60](#).

With externally linked media, if you need to change content—for example, by shortening a sound file—you simply change the original file. As long as the file has the same name and the same location, Authorware automatically includes the change when the piece is played. You can link content over a network or across the Internet. If you edit content frequently in the application you used to create it, linking to external media files is an efficient way to update changes.

Although linking helps keep your piece small, it also means that you end up with lots of separate files to distribute. Linked media files in different locations can be tricky to manage. The External Media Browser helps you manage these files. See [“Using external content and the External Media Browser” on page 105](#).

With embedded media, the content becomes part of your Authorware piece; it is no longer linked to the program that created it. Embedded media is harder to update. To change sound, for example, you must open the original sound file, make the changes, and then reimport the file into Authorware. On the other hand, using embedded media makes file management easier: Although the size of the piece increases because the media files are all contained within it, you need to distribute fewer files.

If you use the same media elements over and over in a piece, you can collect them in libraries. For more information, see [“External content, libraries, and models” on page 103](#).

Embedding and linking content—step-by-step procedures

What you can do	In Authorware Help, see
Import text	Creating and integrating text
Import graphics	Importing a graphic, Linking to external files
Import sound files	Importing a sound, Linking to external files
Import digital movies	Importing a digital movie, Linking to external files
Export embedded media	Exporting embedded content
Check the links to all external files	External Media Browser—step-by-step procedures
Fix a broken link	Fixing a link to an external file
Edit a link	Editing the link to an external file
Change a link to a different file	Linking to a different external file
Define a link with a variable or an expression instead of a literal path	Defining a link with an expression, External content, libraries, and models

What you can do	In Authorware Help, see
Relink to a file after changing its format	Relinking to a file after changing its format
Set up a relative path in place of an absolute path	External Media Browser—step-by-step procedures

Working with text

Text in a piece can be used for headings, questions, prompts, button labels, and simple directions—or as the main feature of pieces such as tutorials, help systems, language training, interactive books, reference titles, and so on. For more information, see [“Text—step-by-step procedures” on page 65](#).

You can do the following things with text:

Display text and move text on the screen Use the Text tool to edit text and create short text segments such as questions, comments, and headings. If you already have large portions of text created in a word processor or editor, you can import the text in rich text format, which retains style information, then edit the text in Authorware. You can change the font, size, and style of text and assign predefined text styles by using the Text menu, or you can add interest by applying transition effects or by moving text along a path or across the screen like a banner.

Use hypertext and hot text Hypertext is text that users click to navigate from one part of a piece to another. Hot text is like a hot spot on a hot object: Authorware reacts in some predefined way when users click it.

Let users enter text Or move text objects as part of an interaction. For example, you can create an interaction that lets users correctly label objects.

You can store text that users enter and set up Authorware so it automatically makes decisions based on the data that users enter. Highly interactive pieces and pieces with instant feedback use this feature.

Send text to and retrieve it from a database You might build a front end to a website that looks up customer records and displays the information inside Authorware. You can have the WriteExtFile function create a record of a student’s performance in a quiz. Or you can use the Authorware Open Database Connectivity (ODBC) capabilities and Structured Query Language (SQL) commands to set up an interface to something such as a referral agency database that lists members, their specialties, and their current availability.

Guidelines for creating and integrating text

Here are some issues that may affect decisions you make about text:

- Not all computers have the same fonts available. If you can’t make sure that the fonts you want to use will get installed on users’ computers, stick with standard system fonts.
- Downloading fonts to Windows computers is clumsy. Although you can use Authorware Web Packager to download the custom fonts that a web-packaged piece needs, Windows users need to restart their computers before the fonts become available. If you don’t know which fonts Windows users have, it’s best to stick with system fonts.

- You can use the Add/Remove Font Resource Knowledge Object to help install and remove fonts. This Knowledge Objects adds a font resource to a user's system by copying the TrueType font you specify to the system director, then loading the font. Before your piece ends, you can use another Add/Remove Font Resource Knowledge Object to remove the font resource from the user's system memory.
- Bitmap text doesn't require text fonts. If you want text with a special look—such as a logo—you can use a bitmap image rather than a text object. Then the text will look the same on all platforms. On the other hand, bitmaps are more difficult to edit, and they use more disk space than editable text objects.

Creating and integrating text

There are four ways to create text and integrate it into an Authorware piece.

Use the RTF Objects Editor to create media-rich RTF documents within Authorware. You can add images, your own graphics, special formatting, hyperlinks, and paste Authorware expressions directly into the RTF Object. You then use an RTF Knowledge Object to include it on the flowline. See [“Using the RTF Objects Editor” on page 66](#).

Creating text within Authorware is a good choice for prototyping, last-minute editing, and short discrete bits of text such as prompts and help messages.



To create or edit text within Authorware, open a display icon or an interaction icon and use the Text tool.

Click in the Presentation window with the Text tool to get the insertion cursor. Click again to create additional text objects. Each text object can be positioned and sized independently of the others.

During editing, a paragraph ruler appears above the text. It has small triangles indicating indent markers and small boxes indicating the text box margins. Click the paragraph ruler to create tab stops; you can drag them to adjust them. Indents apply only to the selected paragraph. Use indented paragraphs to simulate text wrapping around objects.

Importing text is good for long passages of text that appear over many pages and for text that requires specific paragraph formatting, such as hanging indents.

Create text in a word processor and save it in rich text format (RTF). Use the File > Import command in Authorware to bring the text into a display icon. You can then use the Authorware text editing features (text styles or the Text tool and Text menu) to give the text its final form.

RTF text retains all formatting except for headers, footers, footnotes, tables, and columns. Those elements appear as regular paragraph text.

If the text contains page breaks, you can choose whether it is treated as one page or as separate display icons. You can create a text object with or without scroll bars.

Pasting text is another quick way of getting text into your Authorware piece. If you set up paragraph styles in a word processor and you need to add properly formatted text to Authorware, you can create it quickly in a word processor and then copy and paste it into Authorware. For example, an outline would retain all its styles and indents.

Open the document that you want to paste from, select the text to be pasted, and copy it to the Clipboard.

Dragging and dropping a text file directly from your desktop and dropping it on the flowline automatically embeds it in a display icon. Authorware also gives you the option of displaying the text in the file in a scrolling field or of creating a separate display icon for each page of text in the file. It's a great way to set up a paging structure.

Making the most of text

Here are some ways to use text efficiently and creatively.

Paging or scrolling? If you have a block of text that won't fit on a single screen, you have the option of making a scrolling text window or placing the text in another page. With Authorware it's easy to do either.

From a user's point of view, scrolling through text is fine when only a few lines don't fit on the screen. However, scrolling through many screens can be tiring for the eyes.

If you think the best solution is a paging structure where users click a button to advance to the next page, you can use the framework icon to quickly create a book metaphor. Be sure to let users know how many pages of text are to come—or number the pages so users will know where they are.

Assigning text styles You may want to assign specific text attributes—font, size, style, color, even interactivity—to text. Text styles make text easy to read, indicate its purpose, and look attractive. To conveniently format and update entire portions of text, define text styles so you can quickly apply sets of specific text attributes.

Styles help you make text formatting consistent throughout a piece, regardless of the text's source. Modifying a text style's attributes automatically updates the attributes for any text assigned that particular text style.

Searching for text Not only can you search for text as you are authoring (select Edit > Find), but users can search for text as the piece is running. Authorware has a built-in search feature to accommodate searches within the entire piece or within the current framework. To set up a text search, use the navigate icon and set the destination to Search. Users can search for words or keywords assigned to icons. See [“Search—allowing users to search for a page” on page 145](#) for more details.

Controlling text with variables and functions

Authorware has a set of variables and functions designed especially for working with text. The following tables list the variables and functions and give a brief description of each.

For more information about the variables in the following table, see System variables in Authorware Help.

Variable	Description
CapsLock	CapsLock is TRUE if the Caps Lock key is currently on.
CharCount	Contains the number of characters the user typed in response to a text interaction.
EntryText	Contains the text of the user's last response in the last interaction.
ForceCaps	When set to TRUE, all text the user types in the current interaction display is converted to uppercase letters.

Variable	Description
HotTextClicked	Contains the hot text that a user has clicked.
PresetEntry	Text assigned to this variable appears automatically in a text-entry field during presentation.
WordClicked	Contains the word the user clicked in a text object.
WordCount	Contains the number of words the user enters in a text-entry interaction.

For more information about the functions in the following table, see [System variables in Authorware Help](#).

Function	Description
Capitalize	Returns the string that results from capitalizing the first character of every word in the string.
Char	Returns the character, number, symbol, or key name that corresponds to the ASCII code or key name you specify.
CharCount	Returns a count of the number of characters, including spaces, found in a string.
Code	Returns the ASCII code that corresponds to the character, number, symbol, or key name you specify.
DeleteLine	Returns the string that results from deleting a line in a string.
Eval	Makes it possible to evaluate expressions at runtime, based on information the user enters.
EvalAssign	Interprets an assignment operation in a text string as if it were a literal assignment operation.
Find	Searches a text string for the text you're looking for.
GetLine	Retrieves a specific line from a text string.
GetNumber	Returns a numeric value from a series of values in a string.
GetWord	Returns the nth word from a string.
InsertLine	Inserts a line in a string and returns the result.
LineCount	Returns the number of lines in a string.
LowerCase	Converts all the uppercase letters in a string to lowercase letters and returns the result.
MapChars	Maps the characters in a string from the values for one platform to the values for the other.
NumCount	Returns the count of numbers in a string.
Reduce	Eliminates repeated characters and returns the result.
RepeatString	Duplicates a string a given number of times and returns the result.
Replace	Replaces one text segment with another in a string and returns the result.

Function	Description
ReplaceLine	Replaces one line in a string with another line and returns the result.
ReplaceString	Replaces a portion of string with new text and returns the result.
ReplaceWord	Replaces one word in a string with another word and returns the result.
RFind	Searches a string and returns the position of the last match it finds.
String	Converts data into a string.
Strip	Removes one or more characters from a string and returns the results.
SubStr	Returns a portion of a string.
UpperCase	Converts all the lowercase letters in a string to uppercase and returns the results.
WordCount	Returns the number of words in a string.

Text—step-by-step procedures

What you can do	In Authorware Help, see
Add text to the Presentation window	Working with text
Import text	Importing text
Paste text	Pasting text or graphics
Align text	Changing the font, size, style, or alignment of text
Anti-alias text	Anti-aliasing text
Set the font, size, or style of text	Changing the font, size, style, or alignment of text
Define text styles	Defining text styles
Create a style based on the current format of a block of text	Creating a text style from an existing format
Create hypertext	Creating a hot text style, Navigating with hot text, Show Me: Hot Text
Apply text styles	Applying text styles
Modify text styles	Modifying text styles
Change margins	Setting paragraph attributes
Set tabs	Setting paragraph attributes
Indent a paragraph	Setting paragraph attributes
Align paragraphs	Setting paragraph attributes
Map Macintosh fonts to Windows	Font mapping and Macintosh files

What you can do	In Authorware Help, see
Let users enter text	Setting up a text-entry field
Set the style of the text the user enters	Setting the style of user-entered text
Check spelling and add dictionaries	Spelling
Find and change words	Find
Apply transitions to text	Choosing a transition

Using the RTF Objects Editor

The RTF Objects Editor is a powerful tool that lets you create full-featured screens external to Authorware and have them show in the Authorware Presentation window.

To use the features provided with the RTF Objects Editor, you will need to use the Create RTF Object Knowledge Object instead of importing the RTF document into a display icon. See Creating an RTF Object Knowledge Object in Authorware Help.

Note: You can also use any simple word processing program, like Word Pad, to create RTF documents. However, you cannot import graphics, draw your own graphics, or paste Authorware expressions into the files.

Definitions

In order to understand how the RTF Objects Editor and RTF Knowledge Objects work, it is important to be clear about the terminology.

RTF document An external RTF document created by the RTF Objects Editor, an Authorware variable containing RTF codes, or a memo field from a database containing RTF codes.

RTF Objects Editor The application you launch from the Authorware Commands menu which creates RTF documents.

RTF Object An Authorware display object created by either the Create RTF Object Knowledge Object or the `rtfCreate()` function. An RTF Object is transparent and responds to Authorware erase issues, such as before next entry, after next entry, and on exit. Multiple instances are supported. You can use the Authorware erase and motion icons on an RTF Object, and you can export an RTF Object to a graphic.

An RTF document is preprocessed before it is turned into an RTF Object. During this preprocessing phase, `RTFObj.u32` replaces embedded variables and expressions and imports external text and graphics.

RTF Knowledge Objects The Knowledge Objects that allow you to create and manipulate RTF Objects in the Authorware Presentation window. Do not confuse an RTF Object with an RTF Knowledge Object. An RTF Object is a display object, and an RTF Knowledge Object is the Knowledge Object that creates and manipulates the RTF Object.

Several examples are provided in the `RTF.a7p Show Me` located in the Authorware program directory folder `ShowMe/RTF`.

Features

The following are some of the things you can do using the RTF Objects Editor. You can seamlessly include these features in your own Authorware applications.

- Provide non-Authorware developers with the ability to create screens. They'll be able to mix and position text, graphics, and other objects and then easily see the results inside the Authorware Presentation window. This helps you to streamline production work, letting Authorware developers concentrate on the code while other professionals focus on creating your screens.
- Give users the ability to vertically scroll large graphics (or graphics mixed with text and other objects).
- Create external text and format it any way you like. Mix boldface, underlines, italics, colors, and fonts in any combination you want. If you've ever tried to use external text before, you'll really appreciate this feature.
- Create many different polygon shapes not available inside Authorware.
- Use Rich Text Format files. You can either embed them or link them to other files, so that inside Authorware you are always showing the latest version.
- Easily insert the current date and time in a variety of formats.
- Embed any Authorware variables and expressions you want. Authorware evaluates them just as it would if those variables and expressions had been entered directly in the Authorware Presentation window.
- Print anything you've prepared with the RTF Objects Editor. This means users can print anything that you show them through the RTF Objects Editor: a lesson to review offline, a certificate of achievement, or anything you want.

Opening the RTF Objects Editor

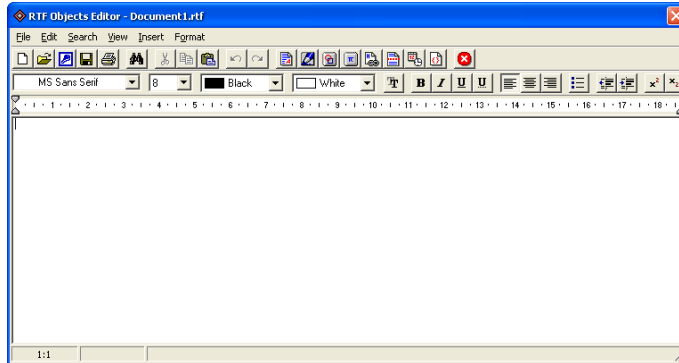
The RTF Objects Editor lets you create rich documents that include formatted text, graphics, and other objects. A non-Authorware user can use it to set up your materials completely outside of Authorware, then easily insert those materials in the Presentation window. Many of the options in the RTF Objects Editor are similar to those in word processors. You can include any of the following objects:

- Text
- Graphics
- Shapes
- Page breaks
- Authorware variables and expressions
- Hot text

Using the RTF Objects Editor, you can open RTF documents and memo fields in Microsoft Access databases and edit them using the RTF Objects Editor features.

To open the RTF Objects Editor:

Select Commands > RTF Objects Editor.



The RTF Objects Editor toolbar

The RTF Objects Editor toolbar contains standard Windows buttons. It also includes several specialized buttons. For information on specific buttons, see [“Managing files with the RTF Objects Editor” on page 68](#) and [“Editing files with the RTF Objects Editor” on page 71](#).

Managing files with the RTF Objects Editor

To create new files and open and close existing files, you can use the commands in the RTF Objects Editor’s File menu. You can also use buttons in the RTF Objects Editor toolbar or keyboard commands for most menu selections.

Opening a new RTF document

When you open a new file, if another RTF document is open, it will be closed first.

To open a new RTF document:

- 1 Select Commands > RTF Objects Editor.
- 2 In the RTF Objects Editor, do one of the following:
 - Select File > New.
 - Click the New Document button in the toolbar.
 - Press Control+N.



Opening an existing RTF document

You can also open an existing RTF document.

To open an existing RTF document:

- 1 In the RTF Objects Editor, do one of the following:
 - Select File > Open Document.
 - Click the Open Document button in the toolbar.
 - Press Control+O.



- A list of RTF documents in the current directory is displayed.
- 2 Select the file you want to open.

Opening a database

You can open an existing Microsoft Access database and view or edit the values in memo fields. The only values you can access are those in memo fields.

You can use the RTF Objects Editor to store RTF information in a database. This is useful because you can store multiple RTF documents in separate database fields, yet deliver a single database with your piece. Otherwise, you would have to deliver multiple RTF documents.

This feature also gives you all the power and capabilities of a database. For example, the Authorware developer could store an entire product catalog in a database, along with a description and a picture of each product. Authorware can be used to browse the database by displaying the previous record, next record, and so on. Storing RTF in database memo fields provides these advantages:

- It makes the product catalog more colorful and interesting for the user.
- It makes building a database browser easier, because the Authorware developer can simply retrieve the fields for the current record and does not need to open external RTF documents in addition to reading the database fields.

To open a Microsoft Access database:

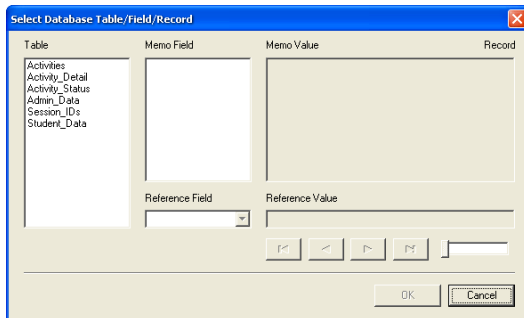
- 1 In the RTF Objects Editor, do one of the following:



- Select File > Open Database.
- Click the Open Database button in the toolbar.
- Press Control+D.

- 2 Select the MS Access Database MDB file you want to open.

The Select Database Table/Field/Record dialog box is displayed.



- 3 Highlight a table to select it.

If a memo field exists in the table, it is displayed in the Memo Field section. The value of the memo field is displayed in the Memo Value section. Reference fields for the memo field are displayed in the Reference field section.

Note: You can edit only the records (references) in memo fields.

- 4 Select a reference field from the Reference Field section.
The Reference fields contain the values for individual records in the database.
 - 5 Do one of the following:
 - To edit a value in a reference field, use the arrows or the slider bar to scroll through the values in the reference field. Display the value you want to edit, then click OK to return to the RTF Objects Editor window and edit it.
 - To insert an image, click OK to return to the RTF Objects Editor window. Select Insert > Image from the RTF Objects Editor menu. Select an image, then click OK.
- When you save the file, the value in the field is updated.

Opening a previously opened RTF document

You can select from a list of the last four opened RTF documents to quickly open a file you have been using.

To open a previously opened RTF document:

- 1 In the RTF Objects Editor, select File > Reopen.
A list of previously opened files is displayed.
- 2 Select the file you want to open.

Saving an RTF document

You can save an RTF document using the same name, which updates the existing file, or you can save a file with a new name, which creates a new file.

To save an RTF document, do one of the following in the RTF Objects Editor:



- To save a file using the same name, select File > Save, click the Save button in the toolbar, or press Control+S.
- To save a file with a new name or location, select File > Save As. Enter the new location and filename.

Printing an RTF document

You can print the RTF document that is currently open.

To print an RTF document:

- 1 In the RTF Objects Editor, do one of the following:
 - Select File > Print.



- Click the Print button in the toolbar.
- Press Control+P.

The Print Setup dialog box is displayed.

- 2 Set the print options as desired.
- 3 Click OK.

Exiting the RTF Objects Editor

To return to the Authorware Design or Presentation window, you need to close the RTF Objects Editor.

To exit the RTF Objects Editor:

1 In the RTF Objects Editor, do one of the following:



- Select File > Exit.
- Click the Exit button on the toolbar.
- Press Alt+F4.

2 If you have not previously saved your file, you are prompted to do so. Click Yes to save your file. You are returned to the Authorware Presentation or Design window.

Editing files with the RTF Objects Editor

You can use the commands in the Edit menu within the RTF Objects Editor as you would the editor in any word processor to select, copy, cut, and paste text and objects.

Undoing the last action

You can return your file to the state it was in before your last action.

To undo the last action, do one of the following in the RTF Objects Editor:



- Select Edit > Undo.
- Click the Undo button in the toolbar.
- Press Control+U.

The file is returned to its previous state.

Redoing the last action

If you have used Undo, Redo will repeat the undone action.

To redo the last action that you undid, do one of the following in the RTF Objects Editor:



- Select Edit > Redo.
- Click the Redo button in the toolbar.
- Press Control+Y.

The file is returned to the state it was in before you used Undo.

Deleting text or graphics

You can delete the currently selected item from a file and copy it to the Clipboard.

To delete text or a graphic and save a copy to the Clipboard:

- 1 In the RTF Objects Editor, select the text or graphic.
- 2 Do one of the following:
 - Select Edit > Cut.



- Click the Cut button in the toolbar.
- Press Control+X.

The selection is deleted from the current location and copied to the Clipboard. You can then paste it into a new location.

Copying text or graphics

You can copy the currently selected item to the Clipboard.

To copy text or a graphic to the Clipboard:

- 1 In the RTF Objects Editor, select the text or graphic.
- 2 Do one of the following:
 - Select Edit > Copy.



- Click the Copy button in the toolbar.
- Press Control+C.

The selection is copied from its current location to the Clipboard. You can then paste a copy into a new location.

Pasting text or graphics

You can paste the contents of the Clipboard at the current insertion point.

To paste material on the Clipboard to a new location:

- 1 In the RTF Objects Editor, select and cut or copy the text or graphic.
- 2 Place the insertion point in the new location.
- 3 Do one of the following:
 - Select Edit > Paste.



- Click the Paste button in the toolbar.
- Press Control+V.

The text or graphic is pasted from the Clipboard to the new location. It remains in the Clipboard and can be pasted again at another location.

Using the Paste Special command

To choose from different formats to use when pasting the current Clipboard contents at the insertion point, you use the Paste Special command. This command allows you to paste text only from an e-mail or another document, without font, color, or other formatting information.

To paste special text or formats:

- 1 Select and cut or copy the text or graphic.
- 2 Place the insertion point in the new location.
- 3 In the RTF Objects Editor, select Edit > Paste Special.
- 4 Select one of the special options (depending on the source):
 - Select Picture if the selection is a graphic.
 - Select Rich Text Format to paste an RTF document with objects.
 - Select Rich Text Format Without Objects to paste an RTF document without objects.
 - Select Unformatted Text to paste only text, with no formatting.
- 5 Select other options as needed:
 - Select Paste Link to paste a link to an existing file.
 - Select Display as Icon to display the pasted item as an icon.
- 6 Click OK.

The text or graphic is pasted from the Clipboard to the new location. It remains in the Clipboard and can be pasted again in another location.

Searching for and replacing text in the RTF Objects Editor

Using the RTF Objects Editor Search menu, you can perform standard search and replace operations.

Searching for text

You can use the RTF Objects Editor to search for any occurrence of text.

To locate text in a document:

- 1 In the RTF Objects Editor, place the insertion point at the place in the file where you want to begin your search.
- 2 Select the text you want to find.
- 3 Do one of the following:
 - Select Search > Find.
 - Click the Find button in the toolbar.
 - Press Control+F.



The Find dialog box is displayed. If you selected text in step 1, the Find What text box displays the text you selected.

- 4 If you did not select text or want to find other text, type it in the text box.

- 5 If you want, select the following options:
 - Select Match Whole Word Only to match only the entire word in the Find What text box.
 - Select Match Case to match the word only when it is in the same case as that in the Find What text box.
 - Select either Up or Down to specify the direction for the search.
- 6 Click Find Next to locate the next occurrence of the text in the direction you selected.
The RTF Objects Editor searches through the file from the insertion point.
- 7 Continue clicking Find Next until you have found all the text you want, then click Cancel.
- 8 To repeat the last Find or Search operation, select Search > Search Again.

Replacing text

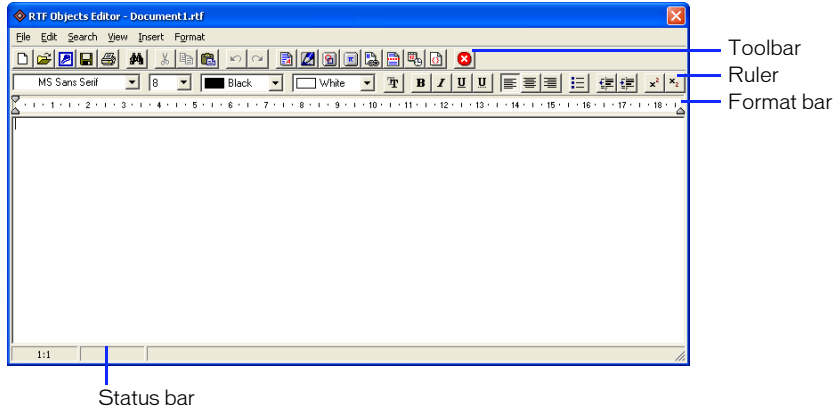
You can use the RTF Objects Editor to search for and replace occurrences of text.

To replace text in a document:

- 1 In the RTF Objects Editor, place the insertion point at the place in the file you want to begin to replace text.
- 2 If you want, select the text you want to replace.
- 3 Select Search > Replace.
The Find and Replace dialog box is displayed. If you selected text in step 2, the Find What text box displays the text you selected.
- 4 If you did not select text or you want to find other text, type it in the text box.
- 5 In the Replace With text box, enter the text you want to substitute for the found text.
- 6 If you want, select the following options:
 - Select Match Whole Word Only to match only the entire word in the Find What text box.
 - Select Match Case to match the word only when it is in the same case as that in the Find What text box.
 - Select either Up or Down to specify the direction for the search.
- 7 Do one of the following:
 - Click Find Next to locate the next occurrence of the text in the direction you selected.
 - Click Replace to replace the first occurrence of the Find What text with the Replace With text.
 - Click Replace All to replace all occurrences of the Find What text with the Replace With text.
The RTF Objects Editor searches through the file from the insertion point.
- 8 Continue clicking the desired buttons until you have replaced all the text you want, then click Cancel.
- 9 To repeat the last Find or Search operation, select Search > Search Again.

Changing view options in the RTF Objects Editor

You can select the toolbars you want displayed in the RTF Objects Editor screen, including the main toolbar, Format bar, ruler, and Status bar. The default is to display all toolbars.



To change the display of toolbars:

- 1 In the RTF Objects Editor, select View.
A list appears showing the toolbars you can display. A check mark precedes those toolbars that are displayed.
- 2 Click each toolbar to select or deselect it, until you have selected the toolbars you want to display.
When you return to the RTF Objects Editor, the selected toolbars are displayed.

Inserting files and graphics using the RTF Objects Editor

You can use the RTF Objects Editor to insert graphics and files directly into your RTF documents.

Inserting a file or image into an RTF document

You can embed or link to an external text (TXT) file, Rich Text Form (RTF) file, or Knowledge Display (KDD) file. You can also embed or link to an external image file, including BMP, JPG, WMF, or EMF formats.

To insert a file or image in an RTF document:

- 1 Use the RTF Objects Editor to open a file.
- 2 In the open RTF document, place the insertion point at the location where you want to insert the file.
- 3 If you are inserting a file, select Insert > File or click the Insert File button on the toolbar.
If you are inserting an image, select Insert > Image or click the Insert Image button on the toolbar.
- 4 Select the type of file you want to insert.
- 5 Select the path and name of the file.

- 6 Select one of the following options:
 - Import File copies the current information from the file into the open RTF document. If changes are made to the imported file after insertion, they are not reflected in the RTF document.
 - Link File copies the location of the file into the open RTF document. Every time the RTF document is opened, the linked file is copied into the RTF document. If changes are made to the linked file, they are reflected in the RTF document.
- 7 If you selected Link File in step 5, select a path option:
 - Absolute (Full Path) looks for the linked file at the exact location (including drive) specified. For example, if you select c:/rtflink for the location of the file, the RTF Objects Editor will always look for the file at that location.
 - Relative to File Location looks for the linked file in a location relative to where the RTF document is currently located. For example, if you select c:/rtflink for the location of the file, but you move the RTF document to the D drive, the RTF Objects Editor will look for the file at d:/rtflink.
- 8 If you are inserting an image, select the Image Preview option you want to use to review the file:
 - Normal displays a preview of the image at its normal size and location.
 - Center displays a preview of the image at its normal size, centered in the Image Preview box.
 - Stretch displays a preview of the image stretched to fill the Image Preview box.
- 9 Click OK to insert the file or image.

Inserting a shape in an RTF document

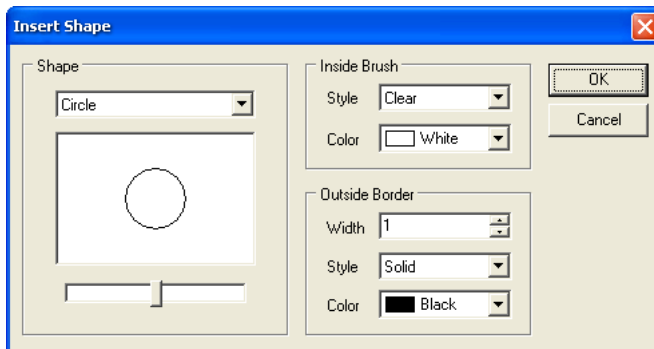
You can create a shape in an RTF document, such as a circle, square, or rectangle, and format it in several ways. This is useful when you use the RTF Objects Editor to create a user interface.

To insert a shape in an RTF document:

- 1 Use the RTF Objects Editor to open an RTF document, or start a new file.
- 2 Place the insertion point at the location where you want to insert the shape.
- 3 Select Insert > Shape or click the Insert Shape button in the toolbar.



The Insert Shape dialog box appears.




- 4 Select the shape you want to insert.

- 5 Change the Inside Brush settings as needed to determine the fill for the shape.
As you change the settings, you see the result in the Shape section of the dialog box.
- 6 Change the Outside Border settings as needed to determine the color, line width, and line style of the border defining the shape.
- 7 Click OK.
The shape is inserted into your RTF document.

Inserting a symbol in an RTF document

You can select and insert a character from any installed font in your RTF document. It's a good idea to use commonly installed fonts, because the end user must also have the font installed in order to see the symbol displayed.

To insert a symbol in an RTF document:


- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Place the insertion point at the location where you want to insert the symbol.
-  3 Select Insert > Symbol or click the Insert Symbol button in the toolbar.
The Insert Symbol dialog box displays.
- 4 Change the font as needed.
- 5 Click the symbol, then click OK to insert it.

Inserting a hyperlink (hot text) in an RTF document

You can insert an Authorware hypertext link, hot text, into an RTF document. When you insert Hot Text into the RTF document, all that is inserted is a `HYPERLINK` tag. This doesn't do anything unless the RTF document is displayed inside Authorware using either the Create RTF Object Knowledge Object or the `rtfDisplay` function. This parses the RTF document and removes the `HYPERLINK` tags.

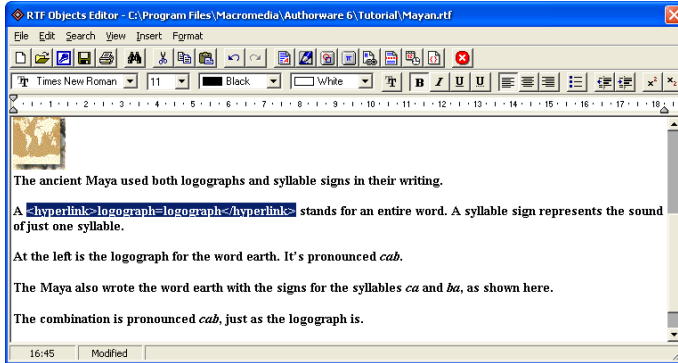
Internally, `RTFObj.u32` remembers the hyperlink positions. You can use this and turn the hyperlinks into something active by either inserting the RTF Hot Text Knowledge Object or using the `rtfGetLinkCount`, `rtfGetLinkCoor`, `rtfGetLinkText`, and `rtfGetLinkCode` functions in a calculation icon. See [“Using RTF functions” on page 83](#) and [“Inserting an RTF Object Hot Text Interaction Knowledge Object” on page 82](#).

To insert a hyperlink in an RTF document:

- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Place the insertion point at the location where you want to insert the hyperlink.
-  3 Select Insert > Hot Text or click the Insert Hot Text button in the toolbar.
- 4 In the LinkText field, enter the text for the hyperlink. For example, if you want the user to click on the words “Go to demonstration”, you would enter **Go to demonstration**.
- 5 In the Link Code field, enter the link code or icon title. For example, if you want to link to an icon titled “Demonstration”, you would enter **Demonstration** in the Link Code field.
- 6 Select Underline Hot Text if you would like the hot text underlined.

7 Click OK.

The hot text is inserted in the RTF document as a hypertext link.



8 The hypertext link will not be active until you insert an Insert RTF Object Hot Text Interaction on the flowline. See [“Inserting an RTF Object Hot Text Interaction Knowledge Object” on page 82.](#)

Inserting a page break in an RTF document

When you create a Create RTF Object Knowledge Object for an RTF document, Authorware creates your RTF Object from the StartPage to the EndPage, reading the <page break> tags. The default is to create an object containing all pages. However, if you want to create an RTF Object starting on page 2, you could enter 2 as the start page and the RTF Object would be created starting at page 2. See [“Creating an RTF Object Knowledge Object” on page 82.](#)

To insert a page break in an RTF document:

- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Place the insertion point at the location where you want the page break.
- 3 Select Insert > Page Break or click the Insert Page Break button in the toolbar.



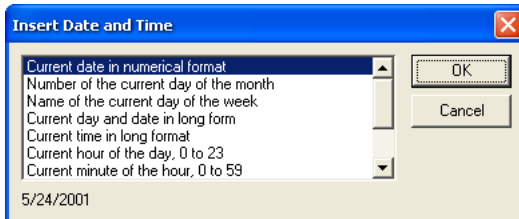
The code <Page Break> appears in the RTF document.

Inserting the date and time in an RTF document

You can insert the date and time in an RTF document, using the Authorware built-in variables.

To insert the date and time in an RTF document:

- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Place the insertion point at the location where you want to insert the date and time.
- 3 Select Insert > Date and Time or click the Insert Date and Time button in the toolbar.



- 4 Select the format for the date and time.
- 5 Click OK.

The date and time are inserted in the file.

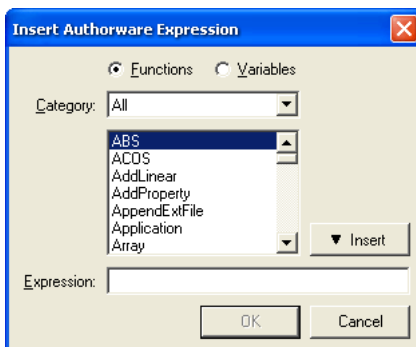
Inserting an Authorware expression in an RTF document

You can insert an Authorware expression in an RTF document. The expression will be executed like any other Authorware expression.

When you insert the expression, the parameters are empty, meaning that default parameters for each function and variable will be used. You can override the defaults by entering the parameters you want to use for the expression.

To insert an Authorware expression in an RTF document:

- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Place the insertion point at the location where you want to insert the Authorware expression.
- 3 Select Insert > Authorware Expression or click the Insert Authorware Expression button in the toolbar.



- 4 Select either Functions or Variables.

- 5 To shorten the list of choices, select the appropriate category.
 - Function categories are All, Character, CMI, Decision, File, Framework, General, Graphics, Icons, Jump, List, Math, Network, OLE, Platform, Target, Time, Video, and [Untitled].
 - Variable categories are All, CMI, Decision, File, Framework, General, Graphics, Icons, Interaction, Network, Time, Video, and [Untitled].
- 6 Select the expression, then click the Insert button.

The selected expression displays at the bottom of the Insert Authorware Expression dialog box.
- 7 Click OK.

The expression is inserted in the RTF object.

Changing formats with the RTF Objects Editor

You can use the Format menu in the RTF Objects Editor to change character and paragraph formatting.

Changing character formatting in your document

You can use the RTF Objects Editor to change character attributes such as font, size, and color.

To change character attributes:

- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Select the text you want to change.
- 3 Select Format > Fonts.
- 4 Select the desired font.
- 5 Set parameters as desired for Font Style, Size, Effects (Strikeout or Underline), Color, and Script.
- 6 Click OK.

Changing paragraph formats

You can use the RTF Objects Editor to change paragraph attributes such as indentation, alignment, and spacing.

To change paragraph alignment:

- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Select the paragraph or paragraphs you want to format.
- 3 Select Format > Paragraph.
- 4 Select parameters as desired for Indentation, Alignment, and Spacing.
- 5 Click OK.

Adding or changing tab stops

You can use the RTF Objects Editor to add, change, and delete tab stops for selected paragraphs.

To add or change tab stops:

- 1 Use the RTF Objects Editor to open an RTF document.
- 2 Select Format > Tabs.
The Tabs dialog box displays.
- 3 In the text box at the top of the Tabs dialog box, type a value or use the arrows to select the position for the tab stop.
- 4 Click Add.
The tab stop is displayed at the bottom of the Tabs dialog box.
- 5 Repeat steps 4 and 5 to add additional tab stops.
- 6 Click Set to set the tabs.
- 7 To clear a tab, highlight it and click Clear.
- 8 When you are finished, click OK.

Using RTF Knowledge Objects

After you have created an RTF document using the RTF Objects Editor, you can add it to the flowline and edit how it functions on the flowline using Authorware RTF Knowledge Objects.

You can use RTF Knowledge Objects in the following ways:

- Use the Create RTF Object Knowledge Object to create a new RTF Object ID variable, which is used in all the other RTF Knowledge Objects to identify an RTF document. See Creating an RTF Object Knowledge Object in Authorware Help.
- Use the Save RTF Object Knowledge Object to export a previously created RTF Knowledge Object to an RTF document or graphic. See Exporting an RTF Object Knowledge Object in Authorware Help.
- Use the Show or Hide RTF Object Knowledge Object to display or hide an RTF Object Knowledge Object when it is running. See Using the Show or Hide RTF Object Knowledge Object in Authorware Help.
- Use the Get RTF Object Text Range Knowledge Object to display the text in a specific range of an RTF document. See Using the Get RTF Object Text Range Knowledge Object in Authorware Help.
- Use the Insert RTF Object Hot Text Interaction Knowledge Object to create an Interaction linked to an RTF document and a specified number of responses. See Inserting an RTF Object Hot Text Interaction Knowledge Object in Authorware Help.
- Use the Search RTF Object Knowledge Object to find a specified text string and return the position of that text string in an RTF document. See Searching for text in a RTF Object Knowledge Object in Authorware Help.

Creating an RTF Object Knowledge Object

After you have created an RTF document using the RTF Objects Editor, you use the Create RTF Object Knowledge Object to provide an RTF Object ID variable name for the RTF document so that you can import it into your Authorware piece.

Note: Do not import an RTF document created using the RTF Objects Editor into a display icon. If you do, the RTF document will not be able to access the RTF Objects Editor features you used to create or edit the file, such as imported graphics, Authorware expressions, or special formatting.

Exporting an RTF Object Knowledge Object

The Save RTF Object Knowledge Object exports a previously created RTF Object to a new RTF document or a graphic after it has been pre-processed. The RTF object to be exported must have been created using the Create RTF Object Knowledge Object.

Using the Show or Hide RTF Object Knowledge Object

The Show or Hide RTF Object Knowledge Object shows or hides an existing RTF object that has been previously created using the Create RTF Object Knowledge Object. Basically this does the same thing as the Show RTF Object option in the Create RTF Object Knowledge Object (see the procedure under Creating an RTF Object Knowledge Object in Authorware Help). The difference is when the object is made visible or invisible.

The Show RTF Object option in the Create RTF Object Knowledge Object allows you to make the RTF Object visible or invisible at creation time. The Show or Hide RTF Object Knowledge Object allows you to make the RTF Object visible or invisible during the execution of the piece, for example, at runtime when a specific condition is met.

You might want to start off with an invisible object and then make it visible later when it is actually needed. Alternately, you might want to start off with a visible object and later hide it. See Creating an RTF Object Knowledge Object in Authorware Help.

Using the Get RTF Object Text Range Knowledge Object

The Get RTF Object Text Range Knowledge Object returns a specified range of text for an existing RTF object that has been previously created using the Create RTF Object Knowledge Object.

Inserting an RTF Object Hot Text Interaction Knowledge Object

The RTF Object Hot Text Interaction Knowledge Object inserts an interaction icon in the flowline with hot spot responses for your RTF object. These responses interact with the hot text in your RTF document. See Inserting a hyperlink (hot text) in an RTF document in Authorware Help.

The hot spot responses created by the RTF Object Hot Text Interaction Knowledge Object use the `rtfGetLinkCoor` function to position themselves over the hyperlinks. When the user clicks a hot spot response, meaning when the user clicks an RTF object hyperlink, the Knowledge Object navigates to the `IconTitle` associated with the hyperlink. This is what is returned by the `rtfGetLinkCode` function. See [“RTF functions” on page 84](#).

However, what you associate with the hyperlink in the RTF Objects Editor doesn't have to be an `IconTitle`. It could be something completely different. Basically, `rtfGetLinkText` returns the visible portion of the hyperlink, and `rtfGetLinkCode` returns the invisible portion of the hyperlink. The RTF Object Hot Text Interaction Knowledge Object assumes the invisible portion is an `IconTitle` and navigates to that `IconTitle`. However, you can modify what is inserted by the RTF Object Hot Text Interaction Knowledge Object, or bypass that Knowledge Object altogether, using the `rtfGetLink` functions in a calculation icon, and do something completely different when the user clicks a hyperlink.

By default, the RTF Object Hot Text Interaction Knowledge Object navigates to an icon, because in practice, this is what most of the users want it to do. That is why the RTF Objects Editor suggests associating an `IconTitle` with the hot text when inserting a hyperlink into the document.

Searching for text in an RTF Object Knowledge Object

You can use the Search RTF Object Knowledge Object to search for specific text in an RTF object that was created with the Create RTF Object Knowledge Object.

Using RTF functions

If you are an experienced Authorware developer, instead of using RTF Knowledge Objects, you can use the underlying RTF functions by themselves. Before you can use them, you need to load `RTFObj.u32`, which contains the RTF functions.

To load RTF functions:

- 1 Select Window > Functions > Load.
The Functions dialog box is displayed.
- 2 Select the category [untitled].
- 3 Click the Load button.
The Load Functions dialog box is displayed.
- 4 Select `RTFObj.u32`, then click Open.
The Custom Functions in `RTFObj.u32` window is displayed, listing all `RTFObj.u32` functions.
- 5 To see a description of a function, click it.
- 6 Select the functions you want to load, then click Load.
The RTF functions are loaded.

RTF functions

When you load the RTF functions, you can use the following functions with RTF objects:

Function	Syntax	Description
<code>rtfCreate</code>	<code>Id := rtfCreate(Left, Top, Width, Height, "Source", Scrolling, Prevent- AutomaticErase, StartPage, EndPage)</code>	<p>This function creates an RTF object with the specified <code>Width</code> and <code>Height</code> on your screen at point (<code>Left</code>, <code>Top</code>) and then reads the RTF document identified by <code>"Source"</code>.</p> <p>This function allows you to set the layer of an RTF Object's position from back to front, relative to other objects on the screen</p> <p>If <code>StartPage</code> and <code>EndPage</code> are supplied, then this function reads the pages from <code>StartPage</code> to <code>EndPage</code>. Otherwise it reads all pages.</p> <p>This function returns the RTF object <code>Id</code> if successful; otherwise it returns 0.</p> <p>Note: This function does not show the object. If you want to display the object in the Authorware Presentation window, you need to call <code>rtfShow()</code> explicitly.</p>
<code>rtfErase</code>	<code>Result := rtfErase(Id)</code>	<p>This function erases the RTF object identified by <code>Id</code>.</p> <p>This function returns 1 (TRUE) if successful; otherwise it returns 0 (FALSE).</p>
<code>rtfFindText</code>	<code>Result := rtfFindText(Id, StartPos, "TextOrPhrase")</code>	<p>This function searches a given range in the RTF object identified by <code>Id</code> for some specific text or a phrase.</p> <p>Use this function to search a range of text in an RTF object for some specific text or a phrase. Only the text in the range starting at the position <code>StartPos</code> will be searched.</p> <p>This function returns the position of the text or phrase, where 0 is the position of the first character in the RTF object. If the text or phrase does not appear in the indicated range, this function returns -1.</p>
<code>rtfGetCurrent- Printer</code>	<code>Result := rtfGetCurrent- Printer()</code>	<p>This function returns the position of the currently selected printer in the list of installed printers, with the first printer having a value of 1.</p> <p>Note: the list of installed printers is returned by the <code>rtfGetPrinterNames()</code> function.</p>
<code>rtfGetLinkCode</code>	<code>Result := rtfGetLink- Code(Id, Num)</code>	<p>This function returns the code of the hyperlink identified by <code>Num</code> in the RTF object identified by <code>Id</code>.</p>
<code>rtfGetLinkCoord</code>	<code>Result := rtfGetLinkCoord- (Id, Num)</code>	<p>This function returns the coordinates of the hyperlink identified by <code>Num</code> in the RTF object identified by <code>Id</code>.</p> <p>Coordinates are relative to the Authorware Presentation window. The return value is a property list that looks like this:</p> <p>[#Left:20, #Top:10, #Width:120, #Height:21]</p> <p>Note: If the RTF object is not visible, this function returns an empty property list.</p>

Function	Syntax	Description
<code>rtfGetLinkCount</code>	<code>Result := rtfGetLinkCount(Id)</code>	This function returns the number of hyperlinks found in the RTF object identified by <code>Id</code> .
<code>rtfGetLinkText</code>	<code>Result := rtfGetLinkText(Id, Num)</code>	This function returns the text of the hyperlink identified by <code>Num</code> in the RTF object identified by <code>Id</code> .
<code>rtfGetOrientation</code>	<code>Result := rtfGetOrientation()</code>	This function returns the currently selected printer orientation: 0: Your RTF document prints vertically on the page (default). 1: Your RTF document prints horizontally on the page.
<code>rtfGetPageCount</code>	<code>Result := rtfGetPageCount("Source")</code>	This function returns the number of pages in the RTF document identified by <code>"Source"</code> .
<code>rtfGetPrinterNames</code>	<code>Result := rtfGetPrinterNames()</code>	This function returns a list of all printers installed in Windows.
<code>rtfGetTextRange</code>	<code>Result := rtfGetTextRange(Id, StartPos, EndPos)</code>	This function returns the text between <code>StartPos</code> and <code>EndPos</code> for the RTF object identified by <code>Id</code> .
<code>rtfHide</code>	<code>Result := rtfHide(Id)</code>	This function hides the RTF object identified by <code>Id</code> . This function returns 1 (TRUE) if successful; otherwise it returns 0 (FALSE).
<code>rtfLineFromPos</code>	<code>Result := rtfLineFromPos(Id, Position)</code>	This function returns the whole line where the specified linear character position appears for the RTF object identified by <code>Id</code> .
<code>rtfPrint</code>	<code>Result := rtfPrint("Source", MarginLeft, MarginTop, MarginRight, MarginBottom, StartPage, EndPage)</code>	This function prints the RTF document identified by <code>"Source"</code> . Margin parameters specify the dimensions, in 1/1000 inches, of the page size used when printing your RTF document. If <code>StartPage</code> and <code>EndPage</code> are supplied, then this function prints the pages from <code>StartPage</code> to <code>EndPage</code> ; otherwise it prints all pages. This function returns 1 (TRUE) if successful, otherwise 0 (FALSE).
<code>rtfSave</code>	<code>Result := rtfSave(Id, FileType, "FileName")</code>	This function exports the RTF object identified by <code>Id</code> to the file specified by <code>"FileName"</code> . <code>FileType</code> can be one of these: 0: RTF 1: BMP 2: JPG 3: GIF This function returns 1 (TRUE) if successful; otherwise it returns 0 (FALSE).

Function	Syntax	Description
<code>rtfSetCurrentPrinter</code>	<code>Result := rtfSetCurrentPrinter(Index)</code>	<p>This function sets the currently selected printer. Index is the printer referenced by its position in the list of installed printers, with the first printer having a value of 1.</p> <p>Note: The list of installed printers is returned by the <code>rtfGetPrinterNames()</code> function.</p> <p>This function returns 1 (TRUE) if successful; otherwise it returns 0 (FALSE).</p>
<code>rtfSetOrientation</code>	<code>Result := rtfSetOrientation(Orientation)</code>	<p>This function determines if your RTF document prints vertically or horizontally on a page. These are the possible values for Orientation:</p> <p>0: Your RTF document prints vertically on the page (default).</p> <p>1: Your RTF document prints horizontally on the page.</p> <p>This function returns 1 (TRUE) if successful; otherwise it returns 0 (FALSE).</p>
<code>rtfShow</code>	<code>Result := rtfShow(Id)</code>	<p>This function displays the RTF object identified by Id.</p> <p>This function returns 1 (TRUE) if successful; otherwise it returns 0 (FALSE).</p>
<code>rtfUpdate</code>	<code>Result := rtfUpdate(Id)</code>	<p>This function updates the RTF object identified by Id.</p> <p>This function returns 1 (TRUE) if successful; otherwise, it returns 0 (FALSE).</p>

Changing the text appearance of RTF objects

RTF objects include the ability to dynamically change font face, size, and style through scripting. You can use the following tags to change text style:

- To make the text bold: `{font-b}...{/font-b}`
- To make the text italic: `{font-i}...{/font-i}`
- To make the text underlined: `{font-u}...{/font-u}`
- To set the font size: `{font-size=18}...{/font-size}`
- To set the font color: `{font-color=red}...{/font-color}`
- To set the font: `{font-face=Arial}...{/font-face}`
- To make the text subscript: `{font-sub}...{/font-sub}`
- To make the text superscript: `{font-super}...{/font-super}`

Aside from setting the font color as shown above, you can also set the font color using hexadecimal color values, such as `{font-color=#FF0000}...{/font-color}`.

When you publish a piece that contains RTF objects for the web, you can either add the RTF documents to the map file or choose to publish them separately. Whichever option you choose, the RTF documents do need to be available on your remote server.

When using `NetLocation()` to open and load your RTF documents, `RTFObj.u32` does not require your RTF documents to be in the map file because `RTFObj.u32` includes a built-in HTTP client and can stream the RTF documents without the map file. When using `FileLocation` to open your RTF documents, you do need to add the RTF documents to the map file.

Note: The Create RTF Object Knowledge Object uses `NetLocation()`, and therefore doesn't require your RTF documents in the map file.

Working with images and graphic objects

The Authorware graphics tools are best used for prototyping and creating simple graphic elements. For more sophisticated graphics you should use a program other than Authorware. Often the graphics that you want to use already exist—for example, a set of engineering schematics, a collection of digitized photographs, or archives of historical documents saved on a CD. At other times you will need the power of a graphics application such as Macromedia Fireworks to create and edit images before you bring them into Authorware. See [“Creating and integrating images and graphics” on page 88](#) for a list of compatible image file formats.

Graphics are placed in display or interaction icons. These icons can hold many graphic objects. You can layer objects and give them various transparency modes. You can group several objects together and treat them as a single object.

Things to consider

Before you start working with images and graphic objects, ask yourself the following questions: “What is the minimum bit depth I can use?” and “What is the minimum bitmap resolution I can use?”

Bit depth If your graphics will be downloaded from the Internet, keep the bit depth to a minimum. A minimum bit depth keeps file sizes small. Saving graphics at a color depth of 8 bits or less improves performance significantly.









Bitmap resolution Most graphics applications let you create bitmaps up to 300 dots per inch (dpi). High resolution such as that is useful for print media, but on the screen most of the information is lost, since monitors usually display around 72 to 96 dpi.

Import graphics at 72 dpi. If you import at a higher or lower resolution, Authorware resizes the graphic. For example, if you import a graphic at 144 dpi, the graphic appears at half its normal size.

Creating and integrating images and graphics

There are several ways to create graphics and integrate them into Authorware. You can use the toolbox to draw graphics, you can import graphics that were created in another application, and you can link to graphics files. You can also cut and paste or drag and drop graphics.

Using the toolbox The toolbox appears when you open a display or interaction icon for editing. Use tools in the toolbox to draw, resize, and rearrange graphics in the Presentation window.

Use this tool	To do this
Pointer 	Select and move objects in the Presentation window. Click an object to select it. Shift-click to select several objects. Drag the center of a filled object to move it. Drag a handle on a selected object to resize it.
Text tool 	Create and edit text. Click in the Presentation window to create a new text object. Click an existing text object to open it for editing. Type in an open text object to edit text.
Straight Line tool 	Draw horizontal, vertical, or 45° lines. Click the start point and drag to draw a line. The line snaps to a horizontal, vertical, or 45° position.
Diagonal Line tool 	Draw a line between any two points. Click the start point and drag to draw the line.
Oval tool 	Draw an ellipse. Click and drag to draw an oval; click and Shift-drag to draw a circle.
Rectangle tool 	Draw a rectangle. Click and drag to draw a rectangle; click and Shift-drag to draw a square.
Rounded Rectangle tool 	Draw a rectangle with rounded corners. Drag the control point that is inside the object to adjust the bevel of the corners.
Polygon tool 	Draw irregular polygons or multiline objects. Click to create the polygon's corners; double-click the last point to complete the polygon. Drag points to modify the line.

Importing images You can import the following types of images: BMP, EMF (Extended Metafile), GIF, JPEG, LRG (xRes), PICT, PNG (Portable Network Graphics), PSD or PS3 (Photoshop), TIFF, TGA (Truevision Targa), and WMF (Windows Metafile). You can't use EMF or WMF images in Macintosh pieces.

Note: To import or play back or view PICT images in Windows, you must have QuickTime 2.0 (or a later version) for Windows installed.

You can preview images before you import them. Once you import an image, select **Modify > Image > Properties** to set the mode (transparent, inverse, and so on), set the background and foreground colors, and view information about the image (file size, file format, and color depth).

Linking to an external image When you select **Import**, the **Import** dialog box gives you the option to **Link to File**. Authorware retains the original file format information, rather than converting to BMP or DIB format. Together with the **External Media Browser**, the **Link to File** option enables you to manage, edit, and substitute images in their original format.

For the advantages and disadvantages of linking to files, see [“Embedding and linking content” on page 60](#).

Making the most of images and graphics

You can set the lines, fills, modes, and colors of objects that you create within Authorware (select **Window > Inspectors**). The **Modes** palette also applies to imported graphics and some digital movie formats. There are six modes: opaque, matted, transparent, inverse, erase, and alpha. There are some movie formats that always play in opaque mode, such as AVI.

Positioning graphics

In Authorware, there are several ways to position graphics and digital movies:

- Enter values (or variables) in the **Layout** tab of the **Properties** dialog box
- Drag objects with the pointer
- Align one or more objects along their top, bottom, left, or right edges; align them horizontally or vertically relative to the center of each object; or align them so they are equally spaced horizontally or vertically or both
- Display a grid and constrain the movement of objects to the grid
- Group several objects to maintain their relative positions as you drag them or to apply an attribute, or for any other reason you might want to treat them as a group

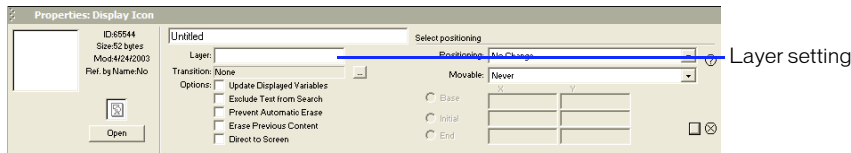
To help position graphics in the **Presentation** window, you can view objects in preceding display icons.

Layering objects

When you arrange several text or graphics objects in a single display icon, some of the objects may overlap. The same thing can happen when you display text and graphics from several different display icons on the screen at the same time. The overlap may cause a problem—for example, a background may hide the text that’s supposed to appear in front of it.

Inside a single display icon Authorware arranges objects from back to front in the order in which you create or import them. To rearrange their order, use the **Bring to Front** and **Send to Back** commands.

In the Presentation window, Authorware arranges objects from back to front in the order in which the icons that contain them appear on the flowline. To move an object to the front or the back—without moving the icon that contains it—use the icon’s Layer setting.



The Layer setting is especially flexible because you can use a variable or expression instead of a number for the setting. With a variable or expression, the order in which objects appear in the Presentation window can change dynamically. For example, you can use variables to set the layering of an object relative to another object. You can also set the layer of an icon’s contents through scripting by using the LayerDisplay function.

Controlling images with variables and functions

Authorware has a set of variables and functions designed especially for working with images. The following tables list the variables and functions and give a brief description of each.

For more information about the variables in the following table, see “System Variables” in Authorware Help.

Variable	Description
DirectToScreen	Draws an image directly to the screen.
DisplayHeight	Contains the height of an image.
DisplayLeft	Contains the distance between the left end of an object and the left edge of the Presentation window.
DisplayTop	Contains the distance between the top of the objects and the top edge of the Presentation window.
DisplayWidth	Contains the width of an object.
DisplayX	Contains the distance from the center of the objects to the left edge of the Presentation window.
DisplayY	Contains the distance from the center of the objects to the top of the Presentation window.
LastX, LastY	Contain the last X and Y coordinates drawn by a graphics function.
Layer	Contains the layer number of the current screen object.

For more information about the functions in the following table, see “System Functions” in Authorware Help.

Function	Description
Box	Draws a box on the screen.
Circle	Draws a circle on the screen.
DrawBox	Allows the user to draw a box.

Function	Description
DrawCircle	Allows the user to draw an oval.
DrawLine	Allows the user to draw a line.
Line	Draws a line on the screen.
Overlapping	Returns a value of TRUE if the graphic objects in the icons you specify overlap.
RGB	Combines the red, green, and blue values into a single color value.
SetFill	Sets the current fill style for drawing functions.
SetFrame	Sets the current frame style for drawing functions.
SetLine	Sets the line style for drawing functions.
SetMode	Sets the current mode for drawing functions.
SetPalette	Loads a palette and sets it as the current palette for the Presentation window.

Graphics—step-by-step procedures

What you can do	In Authorware Help, see
Import a graphic	Importing a graphic, Linking to external files
Preview a graphics file	Importing a graphic
Drag and drop graphics	Drag and drop feature in Authorware
Draw a graphic	Drawing graphics
Cut and paste graphics	Pasting a graphic
Apply a mode, color, fill, or line width	Mode, color, fill, and line width
Blend a graphic's alpha channel with the background	Displaying a graphic's alpha channel
Align graphics	Arranging graphics
Control order in which images appear in front of other images	Placing objects in front of other objects
View graphics in preceding display icons	Displaying the content of several icons
Crop a graphic	Cropping a graphic
Scale a graphic	Scaling a graphic
Use a custom color palette	Using custom color palettes
Move a graphic across the Presentation window	Using the motion icon, Making objects move
Use a special effect to display or erase a graphic	Choosing a transition

What you can do	In Authorware Help, see
Use a graphic as a button	Setting up a hot object, Hot Object Interactions
Synchronize graphics with sound	Media synchronization, SyncPoint, SyncWait functions Show Me: A/V Synchronization
Let the user create graphics	DrawBox, DrawCircle, and DrawLine functions
Use libraries to organize graphics	Using libraries

Working with sound

Sound can be a valuable component of a piece. It can provide information as a narration, enhance the mood of a scene, provide a transition between scenes, or be the focus of the entire piece.

Sound is contained in a sound icon. The sound icon's position on the flowline establishes when the sound plays. The Sound Icon Properties dialog box lets you specify which sound to play and the way it plays.

Guidelines for using sound

Follow these guidelines when you use sound in your Authorware piece:

- It's easier to edit the sound to begin and end where you want before you import it. This is because you must use Authorware functions to specify starting and stopping points of a segment in a sound file, unlike digital movies, from which Authorware can easily select a range of frames.
- Because sound icons are branching icons, they cannot be direct children of interaction, decision, digital movie, or framework icons. However, if you drag a sound icon to an interaction, decision, digital movie, or framework icon or another sound icon, it will automatically be placed in a new map icon.
- Make sure each sound has been recorded at a sampling rate that will give you the sound quality you need for the particular use. The higher the sampling rate of a sound, the better the sound quality but the larger the sound file. For example, music CDs are sampled at 44 kHz, voice-overs may be sampled at 22 kHz, and short sound effects may need no more than 11 kHz. If you're delivering the piece containing the sound over the web, save the sound at an 8-bit sample size and at a sample rate of no more than 11 kHz. Smaller sample rates, such as 5 kHz or 7 kHz, may also be sufficient.
- Mono sounds are significantly smaller than stereo sounds. Unless you require stereo for some specific purpose, always save your sounds as mono to save disk space.
- Compress sounds using Shockwave Audio (SWA), MP3, Voxware (VOX), or ADPCM compression.
- Make sure users' computers have sound cards. A computer that has no sound card can play a piece that includes sound, but the sound itself won't play.
- For 16-bit sounds, make sure users' computers have sound cards capable of playing 16-bit sounds.

File formats for sounds

Authorware supports the following sound file formats: AIFF, MP3, PCM, SWA, VOX, and WAVE.

Creating and integrating sound

Shockwave Audio is a technology that makes sounds smaller and plays them faster off disk or over the Internet. Shockwave Audio can compress sounds up to a ratio of 176-to-1. It also makes them streamable—meaning that they can begin playing before they’re completely loaded into memory. Use Director to create sound files in Shockwave Audio (SWA) format. The higher the bit rate you choose for the compression setting, the better the quality of the sound will be. However, some bit rates are too high for certain Internet access methods. For instance, 16 Kbps, the equivalent of monaural FM radio quality, works for 28.8 Kbps modems, but higher bit rates do not.

You can also convert WAV files to SWA files in Authorware. Select Xtras > Other > Convert WAV to SWA.

Voxware achieves even greater compression for voice-only audio files. When you’re creating simple voice-overs or narration—without music, sound effects, or any background noise—record the audio as a WAVE file, then convert it to the Voxware format (VOX).

If you use an MP3 sound file, be sure to distribute the AWMPP3.x32 Xtra with the file. There is a limit on the types of MP3 files that Authorware will play. If a file is allowed to change bit rates and go from mono to stereo or stereo to mono, there may be a significant change in bit rates. This is a fairly rare condition in most MP3 files, but it is possible. If it turns out that the change in the bit rate is significant enough to throw off the algorithm that imports an MP3 file, then it is possible there will be playback problems with the sound.

The best compression technique and the best compression ratio to use depend on the relative importance of conserving memory and retaining sound quality. Use your judgment. When memory is adequate and high sound quality is the priority, a lower compression ratio or no compression at all is a better choice.

Always make a copy of the original sound before you compress it. After a sound is compressed, you can’t decompress it. You can only change the compression rate by reloading the sound and selecting a different compression rate.

You can import sounds into Authorware by dragging sound files onto the flowline. Authorware creates a sound icon for you. When the sound icon is open, you can preview the sound, choose how fast it plays, set the conditions under which it plays, and set the number of times it repeats.

Synchronizing audio, DVDs, and graphics

Synchronizing multimedia components must be done carefully. Animation, DVDs, and sound each require a certain length of time to play—time that can vary from computer to computer. When you play these components together, making sure they start and stop at the right time in relation to each other is crucial to creating a smooth piece. Synchronizing sound with motion or DVDs usually requires repeated testing to determine how long each component takes to play, where any noticeable pauses occur when a component is loaded into memory, and the start and stop points for each component.

You can place icons to the right of sound or movie icons, creating a *child* icon. In addition, you can set those child icons to trigger based on the media position or number of seconds from when the media starts playing. See Media synchronization in Authorware Help.

Another way to present sound and graphics together is to combine them in a digital movie. This guarantees that the sound and image remain synchronized. It also offers better performance than running a sound icon concurrently with another icon.

A common technique in a presentation is to display text and graphics synchronized with key points in an audio track containing voice-over narration. You can synchronize sounds with visuals by using the `SyncPoint` and `SyncWait` functions.

You can also use the `MediaPosition` variable for synchronization with digital movies such as QuickTime or Video for Windows.

Select Help > Show Me, and then launch A/V Synchronization to see an example of synchronizing sounds with graphics.

Media synchronization

If you place an icon to the right of a sound or movie icon, it becomes a child icon of the sound or movie icon. You can set child icons to trigger based on the media position or number of elapsed seconds from when the media starts playing.

Note: Because sound and movie icons are branching icons themselves, they cannot be direct children of interaction, decision, framework, or other movie or sound icons unless you place them in map icons first.

Controlling sound with variables and functions

You can control sound in your Authorware piece through the use of the following variables and functions. For complete details on how to use the variables and functions, see “System Variables” in Authorware Help.

Variables

For more information about the variables in the following table, see System variables overview in Authorware Help.

Variable	Description
<code>MediaLength</code>	Contains the length of the movie or sound icon you specify. Sound icons return 1000ths of seconds.
<code>MediaPlaying</code>	Determines whether a specific digital movie is playing.
<code>MediaPosition</code>	Tells you the current position of the sound icon.
<code>MediaRate</code>	Contains the rate for the movie or sound icon you specify.
<code>SoundPlaying</code>	Contains the value of <code>TRUE</code> if a sound is playing.

Functions

For more information about the functions in the following table, see “System Functions” in Authorware Help.

Function	Description
MediaPause	Pauses a movie or sound.
MediaPlay	Plays a specific digital movie or sound. When the digital movie is already playing, it is reset to its starting frame and plays from the beginning.
MediaSeek	Sets the position for the movie or sound icon.

Sound—step-by-step procedures

What you can do	In Authorware Help, see
Compress a voice audio file	Compressing voice audio files
Import a sound	Importing a sound, Linking to external files
Preview a sound	Previewing a sound
Set the speed at which a sound plays	Changing a sound's speed
Set the number of times a sound plays	Setting how many times a sound repeats
Integrate a sound with other things that are happening	Creating and integrating sound
Synchronize sounds with graphics	Synchronizing audio, DVDs, and graphics, Media synchronization

Working with digital movies

Digital movies are more than animation and synchronized sound. By adding the ability to select and play specific frames and conserve space through compression, you can:

- Include digital movies created outside Authorware
- Simulate equipment and processes, such as a sprouting seed, readings on a gauge, or varying degrees of rotation for an object
- Compress graphics or sound used in a piece

Authorware gives you added options and possibilities by playing digital movies in response to the conditions you choose. You can play the digital movies when cued by the flowline, when a variable reaches some value, or when the user takes a specific action. By using integers or variables to specify frames, you can play a selected sequence of frames. For specific procedures, see [“Digital movies—step-by-step procedures” on page 98](#).

You can use the Shockwave control for ActiveX to play Director Shockwave movies. In Windows, you can use the Shockwave control for ActiveX to play Director Shockwave movies. ActiveX is not supported on the Macintosh. For information about using ActiveX controls, see Using an ActiveX control in Authorware Help.

Things to consider

Some types of digital movies are always embedded in Authorware; others must be treated as external linked files.

- PICS, FLC/FLI/CEL, and DIB sequence movies are always embedded.
- Director, QuickTime, MPEG, and AVI movies are always external linked files.

Whether the digital movie is embedded or external affects what you must do before delivering it to users.

External digital movies must have the appropriate drivers available and be in the proper format for the platform they play on. For a list of the drivers you must include with each type of digital movie, see [“Determining which files you need to distribute” on page 56](#).

Digital movie file formats

Authorware supports the following digital movie file formats:

- Director
- Video for Windows (AVI)
- Windows Media Player (ASF, ASX, WMV, and IVF)
- QuickTime for Windows (MOV)
- FLC/FLI/CEL
- MPEG
- DIB/BMP sequences

Creating and integrating digital movies

Digital movies are created outside Authorware in applications such as Macromedia Director. Integrating a digital movie into your piece involves placing a digital movie icon in the appropriate place on the flowline, opening the digital movie icon, and selecting a digital movie file. When the digital movie icon is open, you can also preview the movie, choose how fast it plays, and set the number of time it repeats.

To quickly import digital movies, you can drag and drop movies from the desktop into the flowline. Use the Shift key when dragging to link to the movie rather than embedding it.

Because movie icons are branching icons, they cannot be direct children of interaction, decision, framework, sound, or movie icons. You can, however, place a map icon next to the interaction, decision, or framework icon, then place the movie icon inside the map icon.

Making the most of digital movies

Authorware provides you with several ways to use digital movies efficiently and creatively. For information on procedures, see [“Digital movies—step-by-step procedures”](#) on page 98.

Using motion with movies You can use the motion icon with digital movies to move the movie around the screen. For example, you may have an animation of a bird that moves its wings as if it were flying. You can move the bird along a path to simulate the bird flying across the screen.

Playing a range of frames Authorware can play a range of selected frames from a digital movie. The range can be different segments of a digital movie that play at different times, such as different segments from a 360-degree view from the top of Mount Kilimanjaro, or a few frames from a digital movie that simulates turning a knob to different positions. You can identify frames by frame numbers, or you can use variables or other expressions. Using variables or other expressions gives you the flexibility to select different frames at different times, depending on conditions in the piece.

The digital movie can loop continuously, play a set number of times, or play until some specific condition occurs. Use the Movie Icon Properties dialog box to specify the timing and number of times a digital movie plays.

Retaining Director interactivity Unlike QuickTime and Video for Windows movies, Macromedia Director movies are noted for interactivity—the ability to interpret and respond to the user, the computer, or external input. If you already have a Director movie that you want to include in Authorware or if you find that a Director movie can accomplish a task better than Authorware (Director excels in animation and sound), then you will want to take advantage of the ability of Authorware to retain Director interactivity.

Saving Director movies as QuickTime movies If you save a Director movie as a QuickTime movie, you don’t need all the Director driver files and your movie will be much more compact. To save a Director movie as a QuickTime movie, create a movie in Director (without interactivity) and convert it to QuickTime.

Playing AVI movies through the Windows Media Player If you want to play AVI files through the Windows Media Player, instead of the traditional movie icon support, add a WMP extension to the files. For example, you can rename *mymovie.avi* to *mymovie.avi.wmp*. When selected in the digital movie icon, the file plays through the Windows Media Player support. If the DIVX codec is installed on your system (and that of your clients), you can play DIVX movies as well using the same approach.

Turning sound off in digital movies Suppose you want to use a digital movie’s animation but not its sound. You can use the Audio On option to specify whether a digital movie’s sound plays along with the animation.

Seeing objects behind PICS and FLC/FLI movies For PICS and FLC/FLI movies, you can change the layer that the movie appears in. For example, you might want to keep the contents of a previous icon on top and have the contents of subsequent icons appear behind it. You can set the movie’s Layer attribute to make the movie appear in front of or behind other objects regardless of the movie’s place on the flowline.

You might want to let other objects show through. For PICS, FLC/FLI/CEL, and DIB/BMP sequence movies in an Authorware piece, you can do this by applying different modes to the movie. The modes work much the same way as the modes on the modes palette.

Variables and functions for digital movies

Authorware has several variables and functions that track what a digital movie is doing so that you can respond accordingly. The following tables list the variables and functions and give a brief description of each.

For more information about the variables in the following table, see “System Variables” in Authorware Help.

Variable	Description
MediaPlaying	Determines whether a specific digital movie is playing.
MediaPosition	Tells you the current frame number in a digital movie.
MediaLength	Tells you the total number of frames in a digital movie.
MediaRate	Contains the rate for the movie or sound icon you specify.

For more information about the functions in the following table, see “System Functions” in Authorware Help.

Function	Description
GetMovieInstance	Provides a value that serves as an identifier for an instance of a QuickTime or AVI movie within Authorware.
MediaPause	Pauses a movie or sound.
MediaPlay	Plays a specific digital movie or sound. When the digital movie is already playing, it is reset to its starting frame and plays from the beginning.
MediaSeek	Sets the position for the movie or sound icon.

Digital movies—step-by-step procedures

What you can do	In Authorware Help, see
Import a movie	Importing a digital movie
Preview a movie	Previewing a digital movie
Play a range of frames	Playing a range of frames in digital movies
Set a movie’s playback speed	Digital Movie Icon Properties
Set whether a movie plays in front of or behind other images	Specifying a PICS/FLC/FLI/CEL and DIB sequence movie’s layer
Make a movie’s background transparent	Seeing objects behind PICS/FLC/FLI/CEL and DIB sequence movies
Turn off the movie’s sound	Turning off sound in digital movies
Play a sound-only movie	Digital Movie Icon Properties
Set the number of times a movie plays	Setting how many times a digital movie plays

What you can do	In Authorware Help, see
Integrate a movie with other things that are happening	Digital Movie Icon Properties
Position a movie in the Presentation window	Digital Movie Icon Properties
Allow users to move the movie image	Digital Movie Icon Properties
Retain Director interactivity	Allowing Director interactivity
Let users control the playback of a movie	Letting users control QuickTime movies
See objects behind PICS and FLC/FLI movies	Displaying digital movies in front of other objects

Working with DVDs

DVD movies are an excellent source of high-quality images and sound. Training applications where television-quality images are crucial—for example, surgery training in hospitals—can be created by using DVD movies within Authorware. You can set up Authorware to automatically start and stop at specific points within a DVD movie, and you can give students playback controls so they can play, pause, or replay sections if necessary.

For information on how to set up DVD controls, see [“DVDs—step-by-step procedures” on page 102](#).

Things to consider

To play a DVD movie from a DVD drive, you need the following components:

- A DVD that contains an authentic DVD formatted movie
- A DVD drive
- A codec and driver for the DVD drive
- Microsoft DirectX version 8.1 or later

Note: A DVD drive usually comes with a CD that contains the necessary software, drivers, and codecs to run the DVD drive. If your DVD drive did not come with a codec, you might be able to download a free codec from the internet.

Creating and integrating DVDs

Setting up Authorware for DVDs is a three-step process: setting up the hardware, installing the software, and dragging a DVD icon to the Authorware flowline. For information on specific procedures, see [“DVDs—step-by-step procedures” on page 102](#).

Once you are set up for DVD movies, you can set the following playback conditions in the DVD Icon Properties dialog box:

- Which segment to play.
- When to turn the movie or sound on and off.

- When to provide a user control that lets users control the DVD. You might want to allow users to review segments freely in some parts of the course, but withhold control of the DVD during a quiz.
- When to freeze the DVD, so that at the end of a segment, Authorware displays a frame rather than a blank screen.
- Which conditions stop playing the DVD.
- Whether to display the movie in a full screen layout.
- Whether to display captions with the DVD.

Controlling DVDs with variables and functions

The functions and variables related to DVD movies let you precisely control the playing of a DVD movie and access information regarding the DVD movie. The following tables describe the variables and functions.

Variables

For more information about the variables in the following table, see “System Variables” in Authorware Help.

Variable	Description
DVDCurrentTime	Contains the current time, measured in seconds, of the current DVD title.
DVDState	Represents the current playback state of the DVD.
DVDTotalTime	Contains the total time, measured in seconds, of the current DVD title. 0 = the window exists, but the DVD is not playing 1 = the DVD is playing 2 = the DVD is paused 3 = the DVD is scanning forward or backward 4 = the DVD window does not exist
DVDWindowHeight	Contains the height of the DVD window.
DVDWindowWidth	Contains the width of the DVD window.

Functions

For more information about the functions in the following table, see “System Functions” in Authorware Help.

Function	Description
DVDAction	Affects DVD playback based on the symbol specified: #Play = put in play mode #Pause = put in pause mode #Stop = stop playback #End = close window and free all resources #Rewind = put in rewind mode #Fastforward = put in fast-forward mode #Framestep = step to the next frame #Nextchapter = skip to the next chapter #Prevchapter = skip to the previous chapter #Replaychapter = replay the current chapter #Fullscreen = play in full screen mode #Titlemenu = switch to title menu of DVD #Rootmenu = switch to root menu of DVD
DVDCaptions	Turns on the DVD captions when <code>TRUE</code> (if they are present). Specify <code>FALSE</code> to turn captions off.
DVDChapterNum	Returns the number of the current chapter.
DVDCreate	<p>Creates the DVD playback window at the window positions and sizes specified. <code>DVDFilename</code> is the full drive letter, path, and filename of the file to open when creating the DVD window. All of the parameters are optional.</p> <p>If you do not specify both <code>WindowLeft</code> and <code>WindowTop</code> (or they are 0), the window origin will be at 0, 0. If you do not specify <code>WindowHeight</code> and <code>WindowWidth</code> (or they are 0), it creates the DVD window at a size to match the current size of the Authorware presentation window. If you omit <code>DVDFilename</code> (or pass a null string), <code>DVDCreate</code> finds the first DVD disc in a DVD drive on the system and attempts to load that disc.</p> <p>The initial window is created hidden, to allow for audio-only playback. To display the DVD window, use <code>DVDShowWindow(TRUE)</code>.</p> <p>Returns <code>TRUE</code> if successful.</p>
DVDCurrentTitleNum	Returns the current title number.
DVDDirectX	Returns the version number of Microsoft Windows DirectX support installed.
DVDGetDrive	Returns the drive letter of the first DVD drive it finds on the system with a DVD disc loaded. <code>DVDGetDrive</code> begins its search from the drive letter specified. Returns a null string if no DVD drive with a disc is found.
DVDGetVolume	Returns the DVD audio level as a percentage. 100 is full volume, while 0 is no volume.
DVDMute	Mutes the DVD audio when <code>TRUE</code> . Specify <code>FALSE</code> to restore the sound.
DVDNumChapters	Returns the number of chapters in the specified title. If <code>TitleNumber</code> is omitted, it returns the number of chapters in the current title.
DVDNumTitles	Returns the number of titles on the current DVD.

Function	Description
DVDPlayChapter	Plays the specified chapter in the specified title. If TitleNumber is 0, the chapter is played from the current title.
DVDPlaytime	Plays the specified Title number from the specified start time to the specified end time. If Title is 0, the selection plays from the current title number. If the end time is omitted, the playback continues to the end of the title.
DVDSelectButton	Selects the specified button number from the DVD menu.
DVDSetVolume	Sets the DVD audio level as a percentage. 100 is full volume, while 0 is no volume. Returns TRUE if successful; FALSE if not.
DVDShowWindow	Creates the DVD window in a hidden state. Use DVDShowWindow(TRUE) to show the window. Use DVDShowWindow(FALSE) to hide the DVD window if it is displayed.

DVDs—step-by-step procedures

What you can do	In Authorware Help, see
Set up Authorware for DVD	Setting up your computer for DVDs
Preview a DVD segment	Previewing a DVD segment
Select the segment you want to play	Selecting which DVD segment to play
Turn the DVD image off	Turning DVD image and sound on and off
Turn the DVD sound off	Turning DVD image and sound on and off
Provide users with a control panel	Providing a controller
Step a DVD when a certain condition occurs	Turning DVD off

Managing content

Creating content is one of the most time-consuming aspects of any project. Although it's never easy to manage the content you've created, if you do it efficiently you will not only save time, you will end up with a much more effective and professional product.

Here are some benefits of managing content properly:

- You can use the same content more than once without having to duplicate the content.
- You can make content easy to integrate and update, especially if you've used an element such as a graphic or a sound file over and over.
- If you build a flowline structure that solves a common problem or standardizes some functionality—such as a set of custom navigation controls—you can use the structure again elsewhere without have to re-create it.

- You can set up a project so that different team members can share content and flowline structures.
- Content can be easy to update and maintain even after the project has been distributed. For example, you can make new content available over an intranet.

Authorware makes all these things possible through the use of external content, the External Media Browser, media libraries, and models.

External content, libraries, and models

Authorware has four tools that help you manage content:

External content If you link a file to Authorware, the only data Authorware stores is the file's name and its location. The content remains in the file, stored outside Authorware. The great advantage of external content is that it's easy to update both during development and after a project has been distributed. For example, you can turn over a packaged project to a client, and the client—without using Authorware—can update the content just by replacing an external content file with a newer version. Or you can distribute a piece that links to essential but ephemeral content—say, satellite photos of weather systems—over the Internet. As long as the name of the file and its location don't change, you don't need anything to the Authorware piece that the file is linked to.

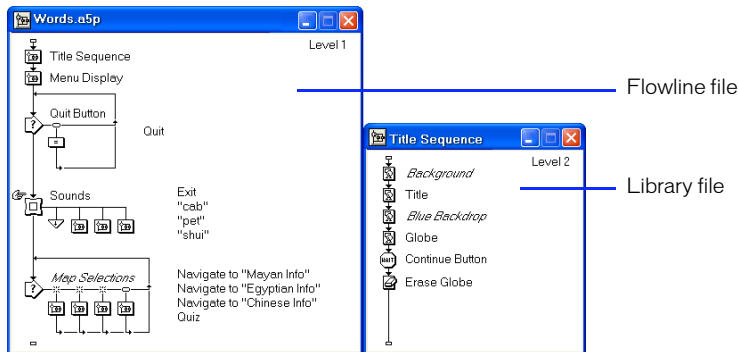
Keep in mind, though, that filenames and locations can't change once you distribute a project. You need to make sure that all the external content files get distributed with the project and installed in the right place or that they get set up correctly on the right web server. Managing external content wisely means knowing when you need its flexibility and when it may well be more trouble than it's worth.

The External Media Browser During the development phase of a project, the External Media Browser makes managing external files easy. It has two major strengths: It lets you edit the pathnames to external files, either individually or in groups, and it lets you define the location of external content with variables and expressions.

At the start of a project you might plan to keep all the external media in a single folder. Later you might decide to place sound, movie, and graphics files in separate folders. You can select all the files in any of the groups in the External Media Browser and, in a single step, change the path to every file in the group.

You can use variables and expressions as paths to change the content of a piece on the fly. Suppose you're creating an information kiosk for a public place such as an airport. The use of external sound files makes the information easy to update, and variables make it possible for users to get the information in the language they choose.

Libraries A library is a special Authorware file that contains a collection of individual icons and their content. Library files are different from Authorware flowline files. Libraries have no flowline and no icon structures—just individual icons.



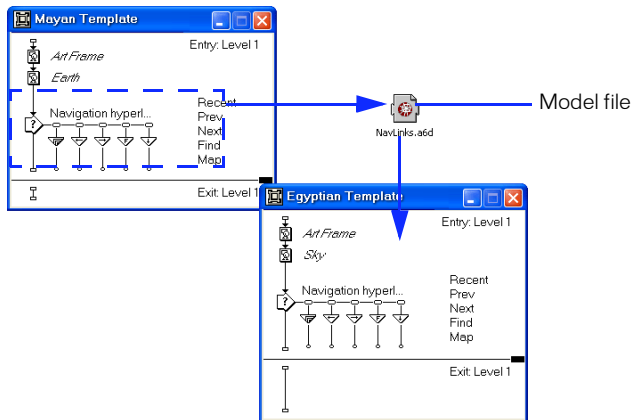
With libraries you can use the same content repeatedly without duplicating it.

Libraries are very much like external content in one way: When you use content that's in a library, the Authorware file stores only a link to the content, not the content itself. What distinguishes a library is that it can contain many different pieces of content, not just one. Because libraries contain icons and not just raw media, they can also contain icon properties; external content files can't contain external properties. For example, to set the start and end frames of a digital movie with a variable, set up the variable in the library. If you were using an external content file you'd have to do it each time you use the movie. Unlike external content files, libraries can be packaged—which means that they're much easier to distribute.

Libraries and external content aren't mutually exclusive. You can link an icon in a library to an external media file as easily as you can link an icon on the flowline to one. The result is the best of both worlds: the content is easy to update, and you have the convenience of setting up the icon properties you want to use just once.

Another advantage of libraries is that they're a convenient development tool for a project team. Once you set up a library, you can place it on a server so that the entire team has access to its contents.

Models A model is a flowline segment—a sequence of icons you’ve copied from the flowline and saved. You can’t open a model directly, but you can reproduce a copy of the icon structure it contains by dragging the model from the Knowledge Objects window to the flowline.



Use models to save and reuse structures you’ve constructed. For example, instead of re-creating a complex interaction, you can just save it as a model. Then you can paste it into any piece you want, and add the text, graphics, and other media that will tailor it to its new context.

Models are unlike both libraries and external content: they’re not a way of saving space. Each time you paste a model on the flowline, you add the complete set of icons that the model contains. But because models are just segments of the flowline, there’s no reason you can’t link icons in a model to external content files or to library icons—as long as the external files and libraries are available to the files you paste the model into.

If you include a Knowledge Object icon in a model, you can turn the model into a Knowledge Object—a packet of logic with its own wizard interface. See [Chapter 11, “Creating Knowledge Objects,”](#) on page 213.

Using external content and the External Media Browser

The great advantage of external media is its flexibility: You can update it as often as you want—even after an Authorware piece has been packaged—without having to make any change to Authorware. Its only drawback is that it requires careful and conscientious management. For every piece of external content in your piece, there’s a file you need to keep track of. The External Media Browser can help you manage external content both during development and when you distribute a project. For information on specific procedures, see [“External Media Browser—step-by-step procedures”](#) on page 106.

Managing external content during development

During development, the External Media Browser provides you with a list of all the external content that’s linked to a file. The browser shows you the name of each external content file, the path to its location, the icon it’s linked to, and whether the link is intact. If you move external content files from one folder to another, you can use the browser to update the paths to the files, either individually or in groups. You can also use the browser to repair links that have been broken.

Using the External Media Browser to prepare for distribution

The External Media Browser makes it easier for you to manage the distribution of external content by letting you change absolute paths to relative paths. An absolute path is the complete path to a file, starting with the hard disk where it's located. Using absolute paths can be problematic. The absolute path to a file on your computer often won't match the absolute path to the same file once it's installed on a user's computer—which means that every time Authorware needs to use the file, it will ask the user to find it. A relative path, on the other hand, is just part of the path to a file, but it always starts from a known reference point: the folder where the Authorware piece is located. As a result, Authorware can find the file without knowing the file's absolute location—and without asking the user to locate it.

The External Media Browser also makes it easier to manage files once you've distributed them by allowing you to locate them with a variable or expression. If you use a variable or expression to identify the external media file for Authorware, you can switch files on the fly. For example, you might substitute a voice-over in one language for a voice-over in another.

Key features

When you're trying to decide how best to use external media, keep these key features of both external content files and the External Media Browser in mind:

- You can update external content files anytime, even after you've packaged an Authorware piece.
- External content files aren't packaged. You can't prevent users from copying them.
- You can use the External Media Browser to change the paths to external content files or to repair broken links.
- You can use the External Media Browser to set relative paths or to locate files with variables or expressions.

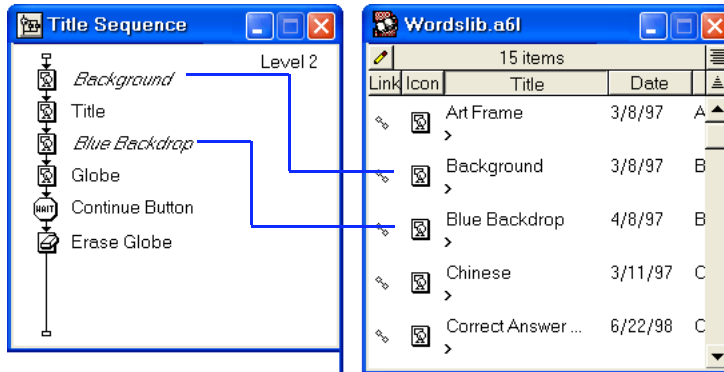
External Media Browser—step-by-step procedures

What you can do	In Authorware Help, see
Check the links to all external files	Using external content and the External Media Browser
Fix a broken link	Fixing a link to an external file
Edit a link	Editing the link to an external file
Change a link to a different file	Linking to a different external file
Define a link with a variable or an expression instead of a literal path	Defining a link with an expression, External content, libraries, and models
Relink to a file after you change its format	Relinking to a file after changing its format

Using libraries

A media library is a special Authorware file that contains a collection of individual icons and their content. You can place display, calculation, digital movie, sound, and interaction icons (but not interaction structures) in a library. For information on specific procedures, see [“Libraries—step-by-step procedures” on page 108](#).

To use a library icon, drag it to the flowline. The icon that appears on the flowline represents a link to the icon and its content in the library.



The titles of icons linked to a library are displayed in italics.

Authorware doesn't actually make a copy of the library icon or its content, it just displays a duplicate image of the icon. The content remains in the library.

Any changes you make to the content of a library icon—for example, adding text or replacing the image there with a different one—are reflected everywhere you use the icon in every file you link to the library.

To organize icons in the library window you can color them, by selecting an icon, then selecting a color from the color palette. You can also sort the icons in the library window by clicking on any of the headings: Link, Color, Icon, Title, or Date.

You can link up to 17 libraries to a piece, and you can have as many as 20 library files open at the same time. (There's an advantage to being able to have more libraries open than you can link to a piece: You can copy icons from the unlinked libraries to the libraries you do want to link to.) Authorware lets you link more than one library to a piece so that you can use libraries to organize content. For example, you might want to put text, graphics, sound, and digital movies all in separate libraries. Or if you're working on a project that's going to be distributed in several languages, each language can be in a separate library.

Note: Don't confuse the button library with a media library. The button library contains only button components, and it's always part of a course file.

Library key features

When you're trying to decide how best to use media libraries, keep these key features in mind:

- With libraries you can use the same content more than once without duplicating it.
- The Authorware file stores only a link to the content, not the content itself.
- You can embed or link content to an icon in a library.
- Unlike an external media file, a library can contain more than one piece of content.
- A library icon can contain icon properties, not just content.
- A library can be packaged, which makes it easy to distribute. But once it's packaged, its content can't be edited.

- You can link as many as 17 libraries to a file.
- An entire team can use a library at the same time.

Libraries—step-by-step procedures

What you can do	In Authorware Help, see
Create a library	Creating, opening, saving, and closing a library
Add icons to a library	Adding icons to a library
Use an icon library in a piece	Using an icon from a library
Delete an icon from a library	Deleting an icon from a library
Paste a library icon into a piece rather than linking to it	Cutting, copying, and pasting library icons
Update the link to an icon in a library	Updating an icon linked to a library
Identify links to library icons	Identifying links to library icons
Fix a broken link	Fixing broken library links
Locate the library an icon is linked to	Identifying links to library icons
List the icons linked to a library	Identifying links to library icons
List broken and unbroken links	Identifying links to library icons
Package a piece and its libraries separately	Packaging a piece and its libraries separately
Package a library	Packaging a library

Using models

A model is a flowline segment—a sequence of icons you’ve copied and saved. A model gives you a much more efficient and reliable way of duplicating a structure than just copying the icons to the Clipboard and immediately pasting them to another location. With a model, you save the structure permanently, so you can use it wherever you want, whenever you want, without returning to the original file where you created it. For information on specific procedures, see [“Models—step-by-step procedures” on page 109](#).

Models are much more powerful than library icons or external media files because they contain chunks of functionality, not just content. For example, you can store a single interaction icon in a library, but to save an entire interaction structure, you need to turn it into a model. Unlike library icons or external media files, models aren’t links to anything. Each time you add a model to the flowline, you’re duplicating the content of the model in its entirety. As a result, you don’t save any disk space by using a model. On the other hand, you can change the structure of the duplicate content to tailor it to the particular context. Changing the structure after you’ve pasted it into a piece has no effect on the model itself.

Models are easy to create and use. To create a model, select the segment of the flowline you want to copy and save it in a model. Put the model file in the Knowledge Objects folder (it’s in the same folder as Authorware). Models in the Knowledge Object folder appear in the Knowledge Objects window. To use the model, drag it from the Knowledge Objects window to the flowline.

Standardizing and sharing elements

Models provide a great way to standardize elements such as buttons and text styles or to share standard components among the members of a development team.

For example, if you want to use the same set of buttons in more than one file, create an interaction that contains all the buttons and then turn the interaction into a model. Drag the model into any piece where you want to use the buttons, save the piece, and then delete the interaction. The buttons remain available in the button library.

If you want to share buttons and text styles among the members of a development team, follow a similar process: Create an interaction with the buttons and use all the text styles in a display icon. Save both as a model and distribute the model to everyone on the team. Any team member can then paste the model into a file, save it, and then delete the interaction and the display icon. The buttons will remain in the button library and the text styles will appear both on the Styles palette and in the Define Styles dialog box.

Add a Knowledge Object icon to a model, and you can create a Knowledge Object—a model with an wizard interface. For example, if you're creating a course with many multiple choice questions, you can create a template interaction and a wizard that allows anyone—even a novice Authorware user—to enter text for and set up a specific interaction.

Model key features

When you're trying to decide how best to use models, keep these key features in mind:

- Models let you create a permanent copy of a chunk of functionality.
- The copy of the model you add to the flowline isn't linked to the model; it's a duplicate.
- Changing the icons contained in the model after you add them to the flowline has no effect on the model.
- Models can contain library icons or linked external content—as long as the libraries or external files are available to the file you paste the model into.
- A development team can use the same set of models to standardize their work.

The complexity of the structure you're copying is another factor in determining whether a model is a useful solution. If the structure is simple, it may be just as easy to re-create it each time you want to use it. If the structure is complex, it may take longer to work out how to modify it than to build a structure tailored to the context in which you want to use it.

In cases where you can easily reuse the functionality you've created, it makes good sense to use a model.

Models—step-by-step procedures

What you can do	In Authorware Help, see
Create a model	Creating a model
Use a model	Using a model
Convert a model from an earlier version of Authorware	Converting a model

CHAPTER 5

Creating Interactions

If you do nothing more than present text, graphics, animation, video, and sound to users, you're creating the equivalent of a television commercial. At best, it will entertain and inform; at worst, users will ignore it. It's a passive, one-way form of communication.

Interactive multimedia holds the key to active, two-way communication. Each interaction provides an opportunity for the user to respond to the program or to other users. You can record and evaluate each response, and you can control which part of the piece the user goes to depending on the response.

You can tailor a multimedia piece to the unique needs of each user. Users can also control the information presented to them. They can navigate through the piece, set the pace, try out procedures, take tests, play games, get further information, and track their own progress. At the same time, you can record information about users: which paths they take, the time they take to respond, the results of tests, what information they receive, and much more.

Macromedia Authorware makes it easy to create a wide variety of interactions with buttons, hot spots, hot objects, hot text, pop-up menus, text-entry fields, and drag-and-drop.

The components of an interaction

Imagine a common computer interaction: A user starts an application, and a list of topics appears on the screen. The user clicks a button to select a topic. The application branches to the segment the user has selected.

These are the basic components of any interaction: a way to interact (a button), a response (a click), and a result (the application branches).

A way of interacting There are many different ways to allow users to interact. You can set up buttons for them to click, menus for them to select from, fields where they can enter text. You can let them click a country on a map to find out more about it or click a word in a passage of text to see its definition. You can wait for them to press a single key to move a pointer around the screen. You can test their knowledge of human anatomy by having them assemble the bones of the skeleton.

When you design an interaction, always select the most efficient way of interacting. For example, it's easier for users to point to a country and click it than to type the name of the country in a text box. But it's quicker for most users to type their name in a text box than to spell it out by clicking letters on the screen.

A response The response is the action the user takes.

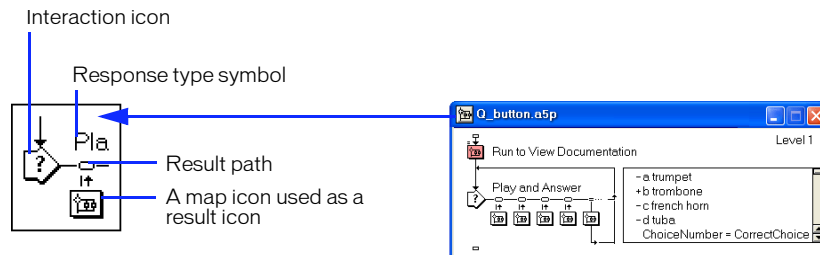
The way you set up an interaction prompts users to respond in a certain way. If you set up a button, you expect users to click it. If you set up a text box, you expect them to type something in it. But users don't always do what you expect. Suppose you've set up a multiple-choice question with answers that are numbered 1, 2, and 3. Most users may click an answer, but others may type 1, 2, or 3. Some may do nothing at all.

Any response that you anticipate a user might make is called a target response. In a multiple-choice question, the right answer is a target response, but so are all the wrong answers. Good design also means anticipating even inappropriate actions, such as typing a number rather than clicking an answer, or doing nothing at all; it means treating the inappropriate actions as target responses and reacting to them.

The result The result of an interaction is whatever you set up Authorware to do in response to a user's action. You might display simple feedback on a wrong answer and a prompt to try again. Or you might launch a web browser and start downloading information over the Internet. The result of an interaction can be anything Authorware can do.

How an interaction works

In Authorware the structure of an interaction looks like this:



Take a look at the four main components: the interaction icon, the response type symbol, the result icon, and the result path. Notice that you can have more than one response type symbol–result icon pair attached to an interaction icon.

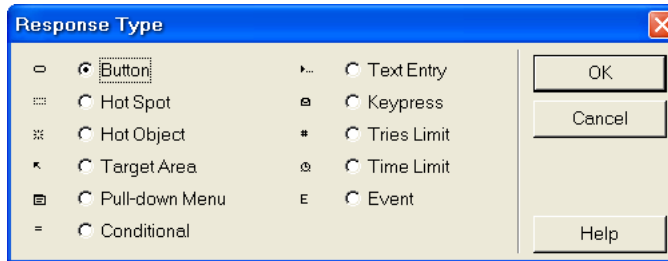
To create an interaction, you drag an interaction icon to the flowline and use it as the foundation of the interaction structure. You build the structure by attaching icons to the interaction icon. Each icon you attach is called a result icon. For each result icon you set up a response type symbol. The response type symbol gives users a way to interact, and it defines the target response that directs the flow down the result path to the result icon. The result path also determines where Authorware goes when it leaves the result icon.

The interaction icon The interaction icon works in part like a display icon: It displays the buttons, menus, text-entry fields, and other elements that allow users to respond to an interaction. You also use it to:

- Display text and graphics that you want to appear throughout an interaction
- Control such things as whether Authorware displays the text and graphics with a transition, where it positions them, which layer it places them in, and when it erases them
- Set whether Authorware erases the screen at the end of an interaction and whether it uses a transition

Most important, however, the interaction icon monitors the actions of the user and sends that information to the response type symbols attached to it.

The response type symbol The response type symbol gives users a way to interact. The response type symbol you select tells the interaction icon whether to display a button, a menu, a text-entry field, or some other element.



Response types and the symbols associated with them

Sometimes you may want to interact not just with users but with circumstances and events. So Authorware provides two other ways to use the response type symbol. One is to use it to track and react to circumstances that are part of the interaction. For example, you can set up a response type symbol to keep track of time and to react if a user doesn't respond within some limit you set. You can also use the response type symbol to track and react to events produced by other components embedded in an Authorware piece, such as an ActiveX pop-up menu or calendar control. For example, you can set up a response type symbol that reacts when the date changes.

Once you've set up a response type symbol to allow a user (or a circumstance or an event) to interact with Authorware, you use the response type symbol to define the target response. Sometimes that's easy (the target response for a button, for instance, is a user's clicking it). Sometimes it's much more difficult than it might seem. Consider, for example, a text-entry field. Suppose you want a user to respond to a question by typing the correct answer. How do you treat a word that's misspelled (*sattellite*, instead of *satellite*), or a synonym (*moon*, instead of *satellite*), or an extra word such as *a* or *the*, or an answer that's so wrong it's impossible to predict? What if you want a number and a user types text? How do you deal with all the numbers that are too large or too small? Authorware has a way of dealing with all those possibilities, but it means anticipating them and setting up target responses to handle them.

As you can guess, most interaction icons have more than one response type symbol attached to them. In some cases the response type symbols represent nothing more than options—the choices on a menu, for example. In other cases, they represent correct and incorrect answers. For that reason, you also use the response type symbol to indicate whether a user's response is right or wrong.

Two final things you do with the response type symbol are related to the result icon: You use it to tell Authorware when to erase any text or graphics displayed by the result icon and to direct the flow after Authorware leaves the result icon.

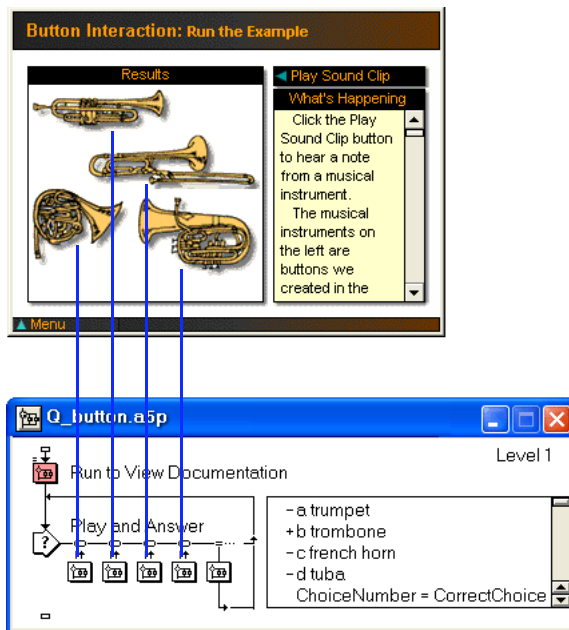
The result icon The content of the result icon—the icon that's attached to the response type symbol—determines what happens when Authorware detects the response type symbol's target response. You can use any icon on the palette as the result icon, but it's best to use a map icon because it's the most flexible. You can add and remove icons from a map icon easily as you construct the result of the interaction.

The result path The result path determines where the flow is directed after Authorware exits the result icon. Typically, you direct the flow to the next response type symbol, back to the interaction icon, or out of the interaction to the next icon on the flowline.

Tracing the flow through an interaction

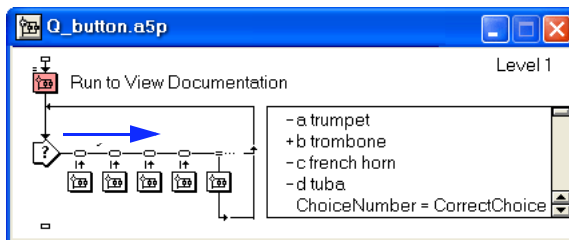
When you're setting up an interaction, you'll find it useful to be able to trace the flow through the interaction structure. If you keep in mind both the structure of the interaction and the way control flows through the structure, it will help you organize the interaction, make sure it's complete, and figure out what's wrong with it if it doesn't work as you expect. For information on specific procedures, see [“Setting up an interaction—step-by-step procedures” on page 117](#).

Waiting for a response When Authorware encounters an interaction icon, it displays any text and graphic elements it finds there as well as any buttons, hot spots, text-entry fields, and so on defined by the response type symbols. It then halts and waits for a response from a user.



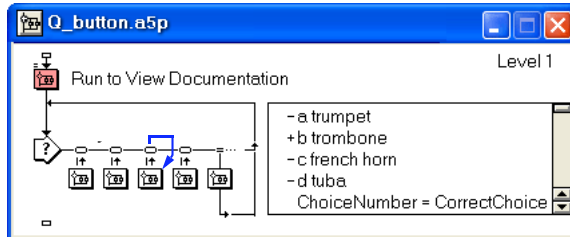
Each response type symbol corresponds to a button.

Checking for a match When a user responds, Authorware sends the response along the interaction flowline to see whether it matches any of the target responses you've set up.



Each target response, you'll remember, is contained in a separate response type symbol. Each response type symbol acts like a switch. If the user's response doesn't match the target response in the response type symbol, the switch stays open, and Authorware sends the response along the flowline to the next response type symbol.

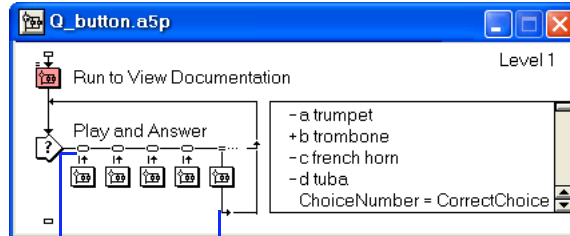
When a user's response matches a target response, the switch closes and directs the flow along the path you've set up for that response.



The flow follows the path to the result icon, where the contents of the icon determine what happens next.

Some target responses are triggered by circumstances rather than a user's action. If you've set up a response type symbol to impose a time limit on an interaction, as soon as the time you've set elapses, the switch closes, and Authorware sends the flow down the interaction flowline. When it reaches the time limit response type symbol, the switch directs it to the result icon that's attached.

Leaving the result icon When the flow leaves the result icon, the result path determines where it goes next. As mentioned earlier, you can select to direct the flow to the next response type symbol, back to the interaction icon, or out of the interaction to the next icon on the flowline.

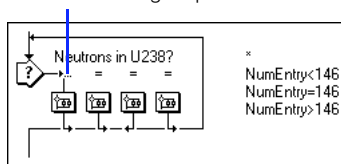


Continue to the next symbol

Try again

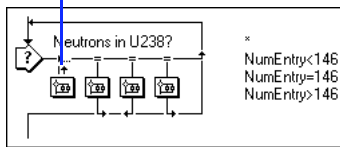
The result path you select and the position of the result icon it's attached to can affect whether or not an interaction works. For example, look at the interaction in the following illustration. The result path that the author selected for the first response type symbol prevents the flow from ever getting to any of the other response type symbols.

The flow never gets past the first response symbol.



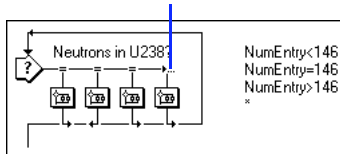
One way to solve the problem is to change the result path so that it goes on to the next response type rather than back to the interaction icon.

The flow continues to the next symbol.

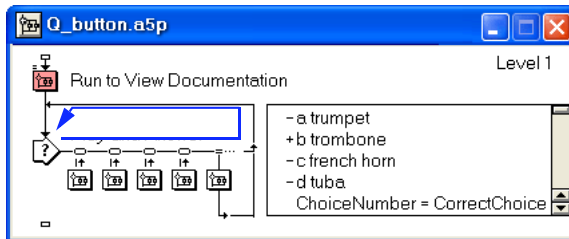


Another way to solve the problem is to make the response type symbol the last one in the structure instead of the first.

The response symbol is now the last one in the structure.



Back to the beginning If the user's response doesn't match any of the target responses in any of the response type symbols, the flow simply returns to the interaction icon, where Authorware waits for another response from the user.



Often it's a good idea to have a catchall target response at the end of an interaction so that it's impossible for a user to respond without getting some feedback. For example, if you set up a multiple-choice question in which users are supposed to click the right answer, you can set up a catchall hot spot that covers the entire screen. If users click anything other than one of the answers, they'll trigger the hot spot. You can give them feedback and then use the result path to take them back to the interaction icon.

Ways to interact

You can give users any of five different ways to interact:

- Click a button, a hot spot, or a hot object
- Select an item from a pull-down menu
- Enter text
- Press a key
- Drag an object to a target

You can also track and react to the following circumstances during an interaction:

- The number of tries a user has made
- The amount of time that has elapsed since the interaction began
- Any condition that can be `true` or `false`

Finally, you can track and react to events produced by components such as ActiveX controls embedded in an Authorware piece.

Each of these ways of interacting is easy to set up with a response type symbol.

Setting up an interaction—step-by-step procedures

What you can do	In Authorware Help, see
Set up a button	Setting up a button, Buttons, radio buttons, and check boxes
Create a pull-down menu	Setting up a pull-down menu, Pull-down menus
Create a hot spot	Setting up a hot spot, Hot spot interactions
Create a hot object	Setting up a hot object, Hot object interactions
Respond to a single key	Waiting for a keypress, Keypress interactions
Create a text-entry field	Setting up a text-entry field, Show Me: Free Form Text, Show Me: Short Text Interaction
Create hot text	Setting up hot text
Create a drag-and-drop interaction	Letting users drag an object, Target area interactions
Limit the number of tries a user has to select a correct answer	Limiting the number of tries, Time limit interactions
Limit the time a user has to respond	Putting a time limit on an interaction, Time limit interactions
React to circumstances that surround the interaction	Reacting to special conditions
Respond to events from sprite Xtra extensions, scripting Xtra extensions, and ActiveX controls	Setting up an event response, Show Me: Using ActiveX
Add text or graphics to an interaction icon	Adding text or graphics to an interaction icon

Buttons, radio buttons, and check boxes

Buttons provide one of the clearest and simplest ways for a user to respond. You can also use the button response type to set up radio buttons and check boxes.



- Use standard buttons when you want the result of a choice a user makes to take effect immediately. Examples are navigation controls, choices on a menu screen, and Help and Quit buttons.
- Use radio buttons for mutually exclusive options. For example, if you're creating a training course that displays text in more than one language, use radio buttons to let users select the language they will use.
- Use check boxes for options that aren't mutually exclusive. For example, if you're creating a kiosk with travel information, use check boxes to allow users to select one or all of the activities they're interested in.

To create the type of button you want to use, you need to set up a button response type symbol and then select either a button, a check box, or a radio button from the Button Library dialog box. If you create a check box or radio button, you can use the system variable `Checked` to find out whether the check box or radio button has been selected.

Button library

The Authorware button library (Window > Buttons) contains standard system buttons, check boxes, and radio buttons for use in setting up a button response type symbol. The button library is just as useful if you're using custom buttons. Assemble the custom buttons in the Button Editor, and they're instantly available in the button library.

Button Editor

Use the Button Editor (Window > Buttons > Add) to create, edit, and globally update custom buttons. In Authorware, a custom button is a complete set of graphics and sounds that represent each of the button's states.

A button can have eight different states—four normal and four checked states, indicating that a button has been clicked, or a check box or radio button has been selected. Each state can look different and have a different sound associated with it. You can also show or hide the button's label in each state.

The four normal button states are as follows:

- The button is ready to be clicked.
- The cursor is over the button.
- The cursor is over the button and the mouse button is down.
- The button is deactivated (unavailable).

Creating a custom button is straightforward. For each of the button's states, you import a graphic and sound and select a setting to show or hide the button's label. If you don't want a different graphic and sound for each state, Authorware automatically uses a graphic and sound you've already imported; or you can select no graphic or sound at all. Before you start, create the graphic or select the clip art image you want to use for each of the button's states. You can create a sound or select a clip sound for each state, too.

Buttons—step-by-step procedures

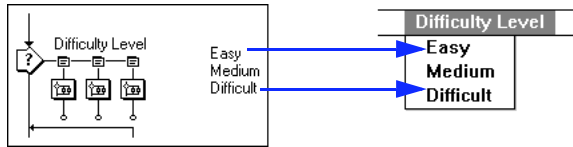
What you can do	In Authorware Help, see
Set up a button	Setting up a button, Show Me: Button Interactions
Adjust the size and position of a button	Resizing or repositioning a button quickly, Resizing or repositioning a button precisely
Use a custom button, check box, or radio button	Selecting a check box, a radio button, or a custom button
Provide a key that a user can press instead of clicking the button	Assigning a shortcut key to a button
Make a button active in the background	Setting up a button that's always available
Make a button active only under certain conditions	Limiting when a button is active
Hide or dim a button when it's not active	Hiding or dimming an inactive button
Make a button the default	Making one button the default
Mark a button as a right or wrong answer	Marking a response right or wrong
Limit the time a user has to click the correct button	Putting a time limit on an interaction
Limit the number of tries a user has to click the correct button	Limiting the number of tries
Create or edit a custom button	Creating a custom button, Editing a custom button
Create a collection of custom buttons	Creating a custom button collection

Pull-down menus

Pull-down menus are ideal for providing users with options without taking up space on the screen. For information on specific procedures, see [“Pull-down menus—step-by-step procedures” on page 120](#).

They're best for options that you want to be available to users over a period of time while they're engaged in some activity, rather than for options that you want to display at a decision point where a user must make a choice. As a consequence, pull-down menus are usually “perpetual” interactions—interactions that are always active in the background no matter what you're doing. For information about perpetual interactions, see [Perpetuals—interactions that are always active in Authorware Help](#).

If you assign shortcut commands to the items on a pull-down menu, Authorware automatically displays the shortcuts on the menu—just as in any standard Windows application.



When you create a pull-down menu interaction, Authorware uses the title you give the interaction icon as the title of the pull-down menu. Unless you enter text, a variable, or an expression in the Menu Item text box in the Response Properties dialog box, Authorware uses the title of each result icon as the name of each item in the menu.

Although all the items in a pull-down menu interaction are usually pull-down menu responses, they don't have to be. For example, you might include a time limit response type symbol that prompts users to make a selection from the menu if they haven't done so after a set amount of time passes.

Pull-down menus—step-by-step procedures

What you can do	In Authorware Help, see
Create a pull-down menu	Setting up a pull-down menu, Pull-down menus
Dim a menu item	Disabling a menu item
Set up the menu to always be available, even when users are engaged in some other activity	Setting up a pull-down menu that's always available
Add a separator between groups of menu items	Adding a separator between groups of menu items
Provide a key that a user can press instead of selecting a menu item	Assigning a shortcut key to a menu item, Assigning a hot key to a menu item
Disable a menu item	Disabling a menu item
Mark a menu item as a right or wrong choice	Marking a response right or wrong
Limit the number of tries a user has to select a given menu item	Limiting the number of tries
Limit the time a user has to select a menu item	Putting a time limit on an interaction

Hot spot interactions

You can make any rectangular area on the screen a hot spot. The area doesn't have to be a discrete graphic, as do buttons and hot objects. That means you can turn an image that's part of a background into a hot spot without editing out the background that surrounds it. Hot spots are also ideal for setting up catchall areas. A catchall area—usually a hot spot as big as the entire Presentation window—lets you respond to users when they click near, not on, the buttons or hot spots you've set up for them to interact with. For specific procedures, see [“Hot spots—step-by-step procedures” on page 121](#).

Hot spots—step-by-step procedures

What you can do	In Authorware Help, see
Create a hot spot	Setting up a hot spot
Adjust the size and position of a hot spot	Resizing or repositioning a hot spot quickly, Resizing or repositioning a hot spot precisely
Provide a key that a user can press instead of clicking a hot spot	Assigning a shortcut key to a hot spot
Set up a hot spot so it's always available	Setting up a hot spot that's always available
Make a hot spot active only under certain conditions	Limiting when a hot spot is active
Use a custom cursor whenever the pointer is over a hot spot	Selecting a custom cursor for a hot spot
Highlight the area that the hot spot encloses when a user clicks the hot spot	Highlighting a hot spot
Set whether you want a user to select a hot spot by clicking it, double-clicking it, or just placing the cursor over it	Clicking, double-clicking, or rolling over a hot spot
Mark a hot spot as a right or wrong choice	Marking a response right or wrong
Limit the time a user has to click the correct hot spot	Putting a time limit on an interaction
Limit the number of tries a user has to click the correct hot spot	Limiting the number of tries
Respond to mouse clicks that aren't over a hot spot	Setting up a catchall hot spot

Hot object interactions

Although a hot object has to be discrete from its background, it doesn't, like a hot spot, have to be rectangular. Making an object hot can be a much more efficient way of making an object interactive than drawing a hot spot over it. It's especially useful if the object isn't rectangular and you want the area that's interactive to correspond to the object exactly—for example, if you want each country on a map to act as a button. It's also useful if you're moving an object around the screen and you want it to remain interactive no matter where it's located.

The most important rule to remember when you're setting up a hot object is to make sure that the object is in a display icon by itself. Authorware actually makes the contents of the entire display icon hot. So if there are several objects in the same display icon, they all become hot and they all react the same way.

For specific procedures, see [“Hot objects—step-by-step procedures” on page 122](#).

Hot objects—step-by-step procedures

What you can do	In Authorware Help, see
Create a hot object	Setting up a hot object
Set up a catchall area around the hot object	Setting up a catchall hot spot
Provide a key that a user can press instead of clicking the hot object	Assigning a shortcut key to a hot object
Set up the hot object so it's available when the user is engaged in other activities	Setting up a hot object that's always available
Make an object hot only under certain conditions	Limiting when a hot object is active
Use a custom cursor whenever the pointer is over the object	Selecting a custom cursor for a hot object
Highlight the object when a user clicks it	Highlighting a hot object
Set whether you want a user to trigger the hot object by clicking it, double-clicking it, or just placing the cursor over it	Clicking, double-clicking, or rolling over a hot object
Mark a hot object as a right or wrong choice	Marking a response right or wrong
Limit the time a user has to click the correct object	Putting a time limit on an interaction
Limit the number of tries a user has to click the correct object	Limiting the number of tries

Keypress interactions

Sometimes the easiest way for a user to do something is to press a single key. For example, if you set up a screen with several text-entry fields on it, the easiest way to get from one field to another is to press the Tab key. You can use a keypress response type symbol to make that possible.

You can set up a keypress response type symbol to react to a single key or to a key combination such as Control-A. You can also set up a single keypress response type symbol to react to several different keys or key combinations (although it can react to only one key or key combination at a time). And you can set up a keypress response type symbol to respond to any key at all, no matter which one a user presses. For specific procedures, see [“Keypress responses—step-by-step procedures” on page 123](#).

Make sure you don't try to use the keypress response type symbol to do things other response type symbols are designed to do. For example, don't use a keypress response type symbol to set up an optional key for a button, hot spot, hot object, or pull-down menu item. Each of those response type symbols has a field in its options dialog box in which you can define an optional key. And don't use the keypress response type symbol when you want a user to enter text in a field. To do that, you use a text entry response type symbol.

Keypress interactions cannot be perpetual. If you want to have a perpetual keypress, you can create a perpetual hot spot response whose area is placed off the screen, and assign a keypress shortcut to the hot spot.

Keypress responses—step-by-step procedures

What you can do	In Authorware Help, see
Respond to a single key	Waiting for a keypress, Show Me: Keypress Interactions
Make a keypress interaction inactive under certain conditions	Limiting when a keypress symbol is active

Text entry interactions

To create a text field where a user can enter text, you set up a text-entry interaction. See [“Text entry—step-by-step procedures” on page 123](#) for information on specific procedures.

Because there’s such a broad range of possibilities to anticipate when you’re allowing users to enter text, Authorware provides considerable flexibility in determining what constitutes a match. For example, you can use wild cards to allow variations in spelling. In fact, you can use wild cards to accept any entry at all. You can also accept different words or phrases that mean the same thing, and you can have Authorware ignore capitalization, extra punctuation, spaces, extra words, or word order.

A text-entry interaction works slightly differently from, say, a button interaction. If you add five button response type symbols to an interaction icon, you get five buttons on the screen. But a text-entry interaction has only one text-entry field, no matter how many text-entry response type symbols you attach to the interaction icon. That’s because no matter how many target responses you set up, a user makes only one entry at a time. If you want to set up several different pieces of information—such as a name and a password—you need to create two separate interactions, one for the name and one for the password.

Authorware lets you evaluate the text a user enters immediately with a text-entry response type symbol, and automatically stores anything a user enters in the variable `EntryText`. You can transfer the text in `EntryText` into a custom variable, then use the information in any way you want—display it on the screen, save it in a file, or write it to a database.

Text entry—step-by-step procedures

What you can do	In Authorware Help, see
Create a text entry field	Setting up a text-entry field, Show Me: Free Form Text, Hot text interactions
Adjust the size and position of a text-entry field	Resizing or repositioning a text-entry field quickly, Resizing or repositioning a text-entry field precisely
Define default text for a text-entry field	Defining the default for a text field
Limit the number of characters a user can enter	Limiting the length of a text entry field
Change the key a user presses to indicate he or she has finished the entry	Changing the entry key
Treat an entry as complete as soon as a user has entered a certain number of characters	Accepting a user’s entry automatically
Ignore an entry if a user presses Enter without entering any text	Ignoring empty text entries

What you can do	In Authorware Help, see
Hide the triangular marker that indicates a text-entry field	Hiding the triangular text-entry field marker
Limit the number of words a user needs to match to accept the entry	Limiting the number of words a user needs to match
Accept several words entered over several tries as a match for the given response	Accepting words on subsequent tries
Use wild cards to allow variations	Text-entry rules
Handle and evaluate numbers	Handling numerical entries
Match any text	Text Entry tab (Text Entry Response Properties)
Select a font, style, and size for the text a user enters	Setting the style of user-entered text
Mark target text as the right or wrong choice	Marking a response right or wrong
Limit the number of tries a user has to enter the right answer	Limiting the number of tries
Limit the time a user has to enter the right answer	Putting a time limit on an interaction

Target area interactions

You can create interesting interactions by letting a user drag an object to a target area. For example, you can ask the user to create a map by dragging countries to their proper location; to assemble a machine by dragging its parts to fit together; to trace the flow of a process by arranging each stage in the correct sequence; or to reconstruct history by lining up events in the order in which they actually happened. You can also create an interaction that gives a user the freedom to landscape a garden or rearrange the furniture in a room. You use the Authorware target area response type for these interactions.

For each target area response type symbol you attach to an interaction icon, Authorware displays a target area on the screen. You then drag the object to connect with the target area to its proper place onscreen. Authorware moves the target area to the location you've indicated.

Target areas—step-by-step procedures

What you can do	In Authorware Help, see
Create a target area interaction	, Letting users drag an object, Target area interactions
Adjust the size and position of a target area	Resizing or repositioning a target area quickly, Resizing or repositioning a target area precisely
Set up a target area so it's always available in the background	Setting up a target area that's always available
Make a target area active only under certain conditions	Limiting when a target area is active
Mark a match as right or wrong	Marking a response right or wrong

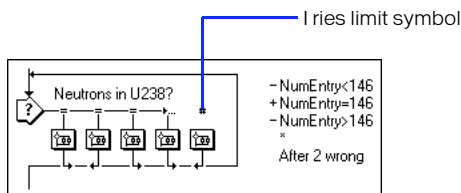
What you can do	In Authorware Help, see
Limit the time a user has to drag an object to the correct target	Putting a time limit on an interaction
Limit the number of tries a user has to drag an object to the correct target area	Limiting the number of tries
Set up a safety net to catch objects that aren't dragged to a target area	Setting up a safety net
Freeze an object you've made movable	Freezing an object you've made movable
Set boundaries for an object that's movable	Setting boundaries for an object that's movable

Tries limit—limiting users' attempts

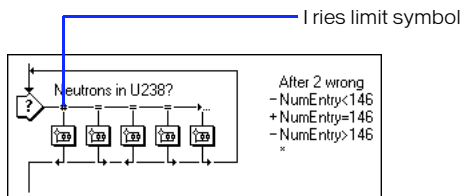
When you set up an interaction that has a right answer, you don't have to select between giving users one chance to get it right and allowing them to try over and over again until they answer correctly. You can use the tries limit response type symbol to limit users to any number of tries.

You can also use the response type symbol to give users feedback after each wrong answer and still let them try again. For example, if a user has three tries to get a question right, you could set a tries limit response to display the message “Two tries left” after the first wrong answer. After another wrong answer, you could set a second tries limit response to display the message “This is your last chance.” Even better, you could give them feedback that could help them figure out the right answer.

Where you position the tries limit response type symbol in the interaction flowline lets you orchestrate the feedback connected to the tries limit symbol and the feedback connected to the other response type symbols. For example, to display the feedback connected to the tries limit response type symbol, you'd position the tries limit symbol to the right of the other symbols, like this:



If you want the feedback connected to the tries limit response type symbol to replace the feedback for the wrong answer, you'd position the tries limit symbol to the left of the other symbols, like this:



Authorware not only uses the tries limit response type to keep track of the number of tries a user has made, it also records the number in the system variable `Tries`. For information on using `Tries`, see “System Variables” in Authorware Help.

Tries limit–step-by-step procedures

What you can do	In Authorware Help, see
Limit the number of tries a user has to enter the right answer	Limiting the number of tries, Tries limit–step-by-step procedures
Use the system variable <code>Tries</code>	Tries
Mark a match as right or wrong	Marking a response right or wrong

Time limit interactions

You can use the time limit response to limit the amount of time a user has to answer a question. You can also use the time limit response to warn users that their time is almost up and to prompt them to respond if they haven't done so after a certain length of time.

When you use a time limit response in an interaction, Authorware uses the system variable `TimeRemaining` to record the amount of time that's left. Authorware also records in the system variable `TimeInInteraction` the amount of time a user has spent in an interaction (even if you haven't used a time limit response in the interaction).

Time limit–step-by-step procedures

What you can do	In Authorware Help, see
Limit the time a user has to enter the right answer	Putting a time limit on an interaction, Time limit interactions
Show the amount of time remaining	Displaying a countdown clock
Restart the time limit for each try a user makes	Restarting the time limit
Make the time limit inactive under certain conditions	Limiting when a time is active

Conditional interactions

Conditional responses are useful to make something happen when some special condition is true. Many variables and expressions that can be true can be used as the target condition for a conditional response.

For example, you may want to check whether the Caps Lock key is down before a user enters text. Or if users have a limited amount of time to complete an entire section of a course, you may want to tell them at the beginning of an interaction how much time they have left. Or you may want to skip an interaction entirely if something is true—for example, if a user has already answered another question correctly.

You can use more than one conditional symbol in an interaction to handle different conditions.

The Automatic setting

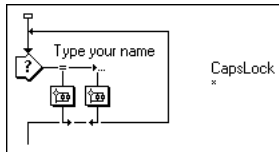
The Automatic setting is a special feature of the conditional response type. It allows a conditional response to react to the target condition in an interaction without the need for a user to respond to the interaction at all. For example, you'd use the Automatic setting to tell Authorware to skip an interaction entirely if a user has already answered another question correctly.

Examples of conditional responses

There's no limit to the ways you can use conditional responses in interactions. Here are a few examples to help you understand the range of situations open to you.

Using a logical variable to check a specific condition

Suppose you want to make sure that the Caps Lock key isn't on when users enter information in a certain text-entry field. You'd set up an interaction that looked like this:



In the Condition field of the Conditional Response Properties dialog box, type:

CapsLock

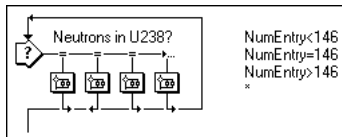
CapsLock is a logical variable. It's true when the Caps Lock key is on and false when the key is off. When the key is on, the target condition is true, so the conditional symbol directs Authorware to the result icon attached to the symbol. The result icon displays a message that tells the user to turn the Caps Lock key off.

Screening out unwanted entries in a text field

Suppose you've set up a text-entry field where you know that entering any number (as in a name field) or entering any text (as in a social security or telephone number field) is a mistake. You can use a conditional response and one of two system variables (NumCount or WordCount) to catch the mistake and ask the user to reenter the information.

Providing feedback for entries in a text field

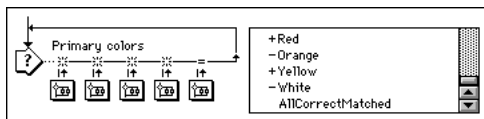
Suppose you want to ask students the question "How many neutrons are there in an atom of Uranium 238?" You want to create a text-entry interaction for their response. The easiest way to set up the interaction is like this:



The target response for the first conditional symbol is any entry that's less than 146. The target response for the second conditional symbol is the correct answer, 146. The target response for the third conditional symbol is any entry that's greater than 146. The asterisk that's used as the target response for the text-entry response is a wild card that accepts any entry.

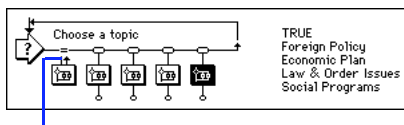
Providing feedback after all the correct items have been selected

Suppose you set up an interaction in which you ask students to select the three primary colors out of six that are displayed on the screen. To set up the structure of the interaction so that students remain in the interaction until they've selected all the correct colors, you use a conditional response type and the system variable `AllCorrectMatched`.



Setting up a condition that's always true

Suppose you've set up a menu, and you want to play the same sound no matter which of the menu items users select. One way to do that is to include a sound icon in the result icon attached to each response type symbol. An easier way is to attach the sound icon to a conditional response type and set up a target condition that's always `true`. The structure of the interaction looks like this:



Continue to the next symbol

By placing the conditional symbol first and selecting Continue as its branching option, you make sure that whenever a user makes a choice, Authorware plays the sound, returns to the interaction flowline, and proceeds to the response type symbol that matches the option the user has selected.

Conditional responses—step-by-step procedures

What you can do	In Authorware Help, see
React to circumstances that surround the interaction	Reacting to special conditions
Set up a conditional response symbol so it's always active in the background	Setting up a conditional response that's always available
Set automatic matching	Conditional Response Properties
Mark a match as right or wrong	Marking a response right or wrong

Events and senders

Sprite Xtra extensions, scripting Xtra extensions, and ActiveX controls can send events. An event is something that happens. Some events are the result of a user's action. For example the Microsoft Calendar ActiveX control tracks where a user clicks on the calendar and sends an event each time that happens. Other events are produced by a sprite, script, or control on its own. The Microsoft Web Browser control, for example, sends an event when it starts to download information from the web and another when the download is complete.

Sprites, scripts, or controls that produce events are called senders. You can use an event response type symbol to track a user's interaction with a sender and respond to it just as if the user were interacting with a button, hot spot, or text entry field you'd set up. You can also set up an interaction to respond to events that a sender generates on its own.

The Event Response Properties dialog box lists all the senders that you've incorporated into an Authorware piece and all the events that each one can generate. All you have to do to set up an event response type symbol is to select the events you want it to track.

Events—step-by-step procedures

What you can do	In Authorware Help, see
Respond to events from sprite Xtra extensions, scripting Xtra extensions, and ActiveX controls	Setting up an event response, ActiveX controls
Make an event response symbol active only under certain conditions	Limiting when an event response is active
Set up an event response symbol so it's always active in the background	Setting up an event response that's always available
Mark a match as right or wrong	Marking a response right or wrong

Perpetual interactions—step-by-step procedures

What you can do	In Authorware Help, see
Make a button active in the background	Setting up a button that's always available
Set up a hot spot so it's always available	Setting up a hot spot that's always available
Set up the hot object so it's available when the user is engaged in other activities	Setting up a hot object that's always available
Set up the menu to always be available, even when the user is engaged in some other activity	Setting up a pull-down menu that's always available
Set up a target area so it's always available in the background	Setting up a target area that's always available
Set up a conditional response symbol so it's always active in the background	Setting up a conditional response that's always available
Set up an event response symbol so it's always active in the background	Setting up an event response that's always available
Turn off a perpetual	Turning off a perpetual
Turn off a perpetual hot spot	Turning off a hot spot perpetual
Make a perpetual inactive	Making a perpetual inactive

Limiting when a choice is active

At times you may want a particular element in an interaction—say, a button or an item on a pull-down menu—to be available only under certain circumstances. To do that, you use the Active If field in the Response Properties dialog box (Modify > Icon > Response). Enter the condition under which you want the element to be available, and Authorware will make it active whenever the condition is `true`.

Limiting when a choice is active—step-by-step procedures

What you can do	In Authorware Help, see
Make a button active only under certain conditions	Limiting when a button is active
Make a hot spot active only under certain conditions	Limiting when a hot spot is active
Make an object hot only under certain conditions	Limiting when a hot object is active
Dim a menu item	Disabling a menu item
Make a keypress interaction inactive under certain conditions	Limiting when a keypress symbol is active
Make a target area active only under certain conditions	Limiting when a target area is active
Make the time limit inactive under certain conditions	Limiting when a time limit is active
Make an event response symbol active only under certain conditions	Limiting when an event response is active

CHAPTER 6

Directing the Flow

The flowline provides a clear image of how Macromedia Authorware operates at its most basic level: it moves forward from icon to icon along a single path.

You'll find that you soon want to move beyond the most basic level. For example, if you're working on a project with many pages of text, you'll want to let users page backward as well as forward, jump back to pages they've already read, and even search for pages that cover a particular topic. Or you may want users to be able to jump to a glossary or a help system without losing their place. If you're setting up a training course, you may want Authorware to branch automatically to a particular section based on the results of a pretest. In any kind of project you're working on, if you find yourself setting up the same sequence of icons over and over, you can set up a subroutine and have Authorware jump to it whenever you need to use it.

Authorware gives you the flexibility to do all those things with its framework, interaction, navigate, and decision icons.

Ways of directing the flow

As you get into the details of directing the flow through an Authorware piece, you'll find that keeping a few key distinctions in mind will prove helpful.

User control versus automatic control

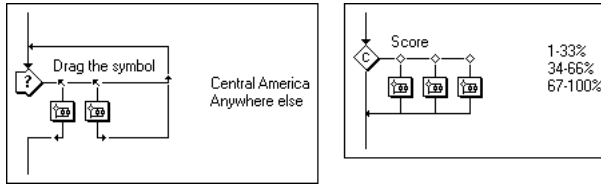
There are two ways to control the flow through a piece:

- You can let users direct the flow. For example, you can provide Next, Previous, and Search buttons to let users navigate through pages of text.
- You can have Authorware direct the flow automatically based on some specific circumstance. In a training course, for example, Authorware might branch automatically to a particular section based on the results of a pretest.

Structures that let users direct the flow are called navigation structures. Structures that direct the flow automatically are called decision structures.

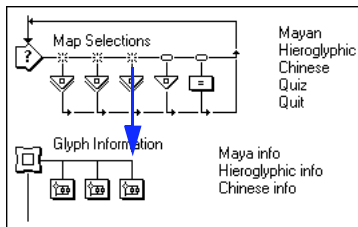
Following a path versus jumping around

In many cases, directing the flow in Authorware is just a matter of channeling it down one path or another. That's the case with interaction icons and decision icons. The flow never leaves the flowline; it simply follows one branch rather than another.



Examples of paths

In other cases, directing the flow means setting up Authorware to jump from one place to another on the flowline. You set up jumps with navigate and framework icons. You can place a navigate icon anywhere in a file, but it can direct the flow only to an icon that's attached to a framework icon.



The best place to use navigate icons is within framework structures. Inside a framework structure you can jump around as much as you like.

The flow within a file versus the flow between files

A project that you're working on may be made up of several files, each containing a different segment. If so, you need to be able to direct the flow from one file to another.

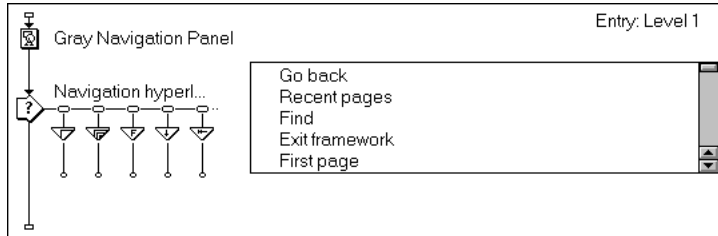
Directing the flow from one file to another is different from directing the flow from one icon to another within a single file. You can't use the navigate icon to jump from one file to another. Instead, you need to use one of the `JumpFile` functions (see [“Using Variables, Functions, Xtra Extensions, and More”](#) on page 157).

The flow and the Internet

If you're designing an Authorware piece to run over the Internet, remember that downloading data over the Internet takes longer than reading the same data from a local hard disk. The strategies you select to direct the flow may need to be different.

Navigate, framework, interaction, and decision icons

Don't think of the navigate, framework, interaction, and decision icons as mutually exclusive ways of directing the flow. They are all useful in various combinations to achieve the kind of control you want. For example, the framework icon has built into it an interaction icon with a series of navigate icons attached to it. Those navigate icons are the framework icon's standard navigation controls.



You could also use a navigate icon—or even an entire framework structure—at the end of each path in a decision structure.

In designing ways to control the flow through an Authorware piece, remember what each icon does:

- Interaction icon lets users select a branch.
- Decision icon selects a branch automatically.
- Navigate icon jumps the flow from one place to another.
- Framework icon contains the destinations the navigate icon can jump to.

Decision structures—directing the flow automatically

A decision structure is made up of a decision icon and one or more paths that are attached to it. When the flow reaches the decision icon, the decision icon automatically directs the flow down one of the paths, based on the settings you've selected for the icon.





Suppose, for example, that you're creating a training course for customer service representatives. Some of the people taking the course are current employees brushing up on department policies, some are new employees unfamiliar with the system, and some are seasonal workers who may have learned the system last year but haven't used it since then. Because the backgrounds of the people taking the course are so varied, you decide to create a pretest to determine where each user should start.

At the beginning of the course, Authorware presents each user with the pretest and then encounters a decision structure on the flowline.

The decision structure tells Authorware which path to follow based on the user's score. That's the essence of a decision structure: The path Authorware follows is neither predetermined nor decided by the user. Instead, the path is determined by Authorware according to conditions you've set when it reaches the decision structure.

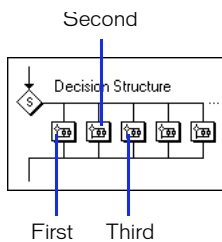
Types of branching

The decision icon lets you select four types of branching:

Decision icon	Type of branching
	Sequential
	Random to any path
	Random to only an unused path
	To a path determined by a calculation

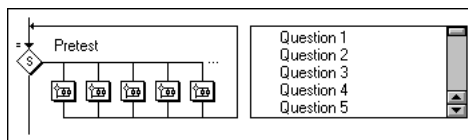
Sequential branching

Sequential branching works like this: Each time Authorware passes through a decision structure, it takes a different path, starting with the one on the left. The second time it passes through the structure, it takes the second path; the third time, it takes the third path, and so on.



The introduction to this section described how you can use a decision icon to evaluate a pretest and take the path to the part of a course where it's appropriate for a user to start. You can also use a decision icon to present a pretest.

You set up the decision structure so that each path presents one of 10 questions in the quiz.



You attach a calculation to the decision icon and, to keep track of the number of questions a user has answered, enter this expression in the Calculation window:

```
counter := counter+1
```

The expression adds 1 to the custom variable `counter` each time Authorware passes through the decision structure.

In the Decision Options dialog box, you select **Until True** as the Repeat option and enter this expression:

```
PercentWrong>30 | counter=10
```

The first expression, `PercentWrong>30`, tells Authorware to continue repeating the decision structure until the user has answered more than 30 percent of the questions wrong. The second expression, `counter=10`, tells Authorware to continue repeating the decision structure until the user has answered all the questions. The “or” (`|`) bar between the two tells Authorware to repeat the decision structure until either expression is true.

Because you’ve set the decision icon to branch sequentially, each time Authorware passes through the decision structure it moves one path to the right. If you don’t include an expression to tell Authorware when to stop repeating the decision structure, when it reaches the last path it returns to the left-most path and starts over again.

The Reset Path On Entry option affects what Authorware does if it leaves the decision structure and then returns to it later. If Reset Path On Entry is selected, Authorware starts over again with the first path. If it isn’t selected, Authorware remembers which path it followed the last time it passed through the decision structure and moves one path to the right.

Random branching

There are two ways to set up random branching in a decision structure. You can have Authorware select any path at random, regardless of whether it’s selected the path before, or you can have Authorware select randomly only from the paths it hasn’t yet taken.

For example, you could set up the pretest described in the previous procedure so that Authorware selects randomly from the questions it hasn’t yet presented a user. Everything else about the decision structure would be the same except that you’d select Randomly To Unused Path rather than Sequentially as the Branch option.

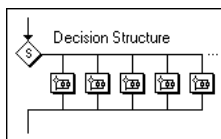
Randomly To Unused Path also comes in handy if you need to test a student a second time on the same material. You’d set up each question so that it has two or more versions, depending on the number of times a student is allowed to retake the test.

If you leave Reset Path On Entry deselected, Authorware remembers which version of the question it has already presented. The next time it passes through the decision structure, it presents a different version.

Branching based on a variable or expression

You can use the value of a variable or expression to determine which path in a decision structure Authorware follows.

Authorware numbers the icons attached to a decision icon from left to right, starting with 1.



If a variable’s value is 1, Authorware branches to the first icon; if the variable equals 2, it branches to the second icon. If the value of a variable isn’t a whole number, Authorware ignores the decimal part of a value; it considers 0.75 equal to 0 and 4.25 equal to 4. Notice that Authorware doesn’t round the value to the nearest whole number—it simply drops the decimal part of the value. If a value is less than 1 or larger than the number of icons attached to the decision icon, Authorware bypasses all the attached icons and moves to the icon that follows the decision icon on the flowline, regardless of which Repeat option you’ve selected.

If you want something to happen only when a certain condition is `true`, use a logical variable or a conditional expression. If the variable or expression is `true`, Authorware treats it as if it's equal to 1 and branches to the first icon attached to the decision icon. If the variable or expression is `false`, Authorware treats it as if it's equal to 0, bypasses the icon attached to the decision icon, and moves to the icon following the decision icon on the flowline. For an example, see [“Branching based on a conditional expression” on page 136](#).

Branching based on a calculation

Suppose, for example, that you're setting up the training course described in [Decision structures](#) (see [“Decision structures—directing the flow automatically” on page 133](#)).

At the beginning of the course, Authorware presents each user with the pretest and then encounters a decision structure on the flowline. Remember, if the value of a variable or an expression isn't a whole number, Authorware ignores the decimal part of the value.

The calculation divides those who take the pretest into three groups. As shown in the following table, the lowest third take the first path, the middle third take the second path, and the highest third take the third path.

Percent Correct	(PercentCorrect/34)+1	Path
25	1.74	1
33	1.97	1
34	2	2
50	2.47	2
67	2.97	2
68	2	3
100	2.94	3

Branching based on a conditional expression

You can use a conditional expression when you want something to happen only if a condition is `true`. Suppose you have text interactions in a piece that are case-sensitive and you want to make sure at the beginning of the application that Caps Lock isn't down. You can use the system variable `CapsLock` to find out. `CapsLock` is `true` if Caps Lock is down, `false` if it's up.

You'd set up a decision structure to display a message to the user if Caps Lock is down. The decision structure has only one path.

If `CapsLock` is `true`, Authorware branches to the display icon attached to the decision icon; if `CapsLock` is `false`, Authorware bypasses the decision structure entirely.

To have Authorware check the value of `CapsLock`, you insert the variable into the To Calculated Path field.

You can also set up branching based on a variable or expression.

Decision structures—step-by-step procedures

What you can do	In Authorware Help, see
Branch sequentially to a series of paths	Setting up sequential branching
Branch randomly to a series of paths	Setting up a decision path
Branch randomly to only to the paths that haven't already been used	Random branching
Branch based on the value of a variable or an expression	Branching based on a variable or expression

Paging structures—quick and easy user control

A paging structure is any navigation structure that uses a book or index card metaphor. A paging structure usually consists of pages of text and has navigation controls that let users go to the next or the previous page, look up and jump to a page they've already read, and sometimes search for a word or a topic and jump to the page where it's located.

Paging structures are the easiest type of navigation structure to create in Authorware. Once you've saved the content in a rich text format (RTF) file, all you need to do is drag a framework icon to the flowline, and then drag the RTF file to the framework icon. Authorware creates a display icon for each page in the RTF file, attaches the display icons to the framework icon, and provides a ready-to-use set of navigation controls. The results aren't elegant, but they're quick and serviceable—and it's a great way to get started or to prototype a project.

The navigation controls built into the framework icon give users these options:

- Move to the next or previous page
- Move to the first or last page in the paging structure
- Jump back to the last page they've read
- Display a list of the pages they've read
- Search for a word, phrase, or keyword
- Exit the paging structure

Although any icon attached to a framework icon is called a page, a page doesn't have to be a screen full of text. It can be a graphic, an animation, a digital movie, a sound—or even a map icon containing a complex sequence of icons. For example, you could use a paging structure to create an annotated version of Chopin's waltzes. Each page might contain a recording of one of the waltzes accompanied by a commentary on the music. You could use a paging structure to show how editing changes a film. For example, you could use the first page to show an unedited scene, the second to show the scene after it's been cut, the third to show with special effects, the fourth with sound effects, the fifth with music, and so on.

By the way, if you drag a group of movie files or a group of sound files—or even a group of movie, sound, graphic, and text files—to a framework icon, Authorware places each of them in its appropriate icon and arranges them as pages attached to the framework icon.

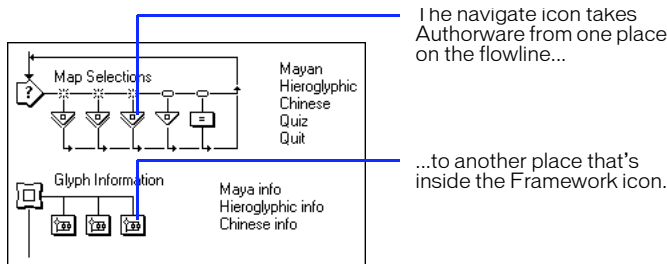
Paging structures—step-by-step procedures

What you can do	In Authorware Help, see
Set up a paging structure	Setting up a paging structure
Set up a paging structure a page at a time	Setting up pages one at a time
Import text into a paging structure from an RTF file	Importing text into a framework
Reposition the default navigation controls	Repositioning the navigation controls
Select a transition effect between pages	Setting up page transitions
Disable Previous and Next buttons so piece doesn't go from last page to first or from first to last	Preventing wraparound paging Going to the next, previous, first, or last page
Change the way a navigation control works	Changing where a navigation control takes a user
Change the look of a navigation control or the sound it makes	Changing a navigation control's graphic or sound
Provide a key that a user can press instead of clicking a navigation control	Assigning a shortcut key to a button
Make a navigation control active only under certain conditions	Limiting when a button is active
Hide or dim a navigation control when it's not active	Hiding or dimming an inactive button
Make a navigation control the default	Making one button the default
Add or delete a default navigation control	Adding or deleting a default navigation control
Create custom navigation controls	Creating custom navigation controls
Save navigation controls as a model	Saving navigation controls as a model
Replace the default navigation controls with a model	Replacing the default navigation controls with a model
Remove the default navigation controls from the framework icon permanently	Eliminating the default navigation controls
Create new default navigation controls	Creating new default navigation controls

Frameworks in depth

Navigation structures let users select the path they want to take through a piece. Most navigation in Authorware is handled with the navigate and framework icons. The two icons work hand in hand.

The navigate icon takes Authorware from one place on the flowline to another place that's inside a framework icon.



A navigate icon can be on the flowline outside a framework, but the destination it directs Authorware to must be attached to a framework.

The navigate icon is much more sophisticated and flexible than the Authorware GoTo function. You can use the navigate icon to do the following:

- Go to any specific icon attached to a framework
- Go to an icon just by identifying its location relative to the current icon (for example, the next icon or the previous one)
- Return to the last icon a user looked at
- Display a list of all the icons a user looked at recently
- Display a dialog box that a user can use to search for an icon and then go to it

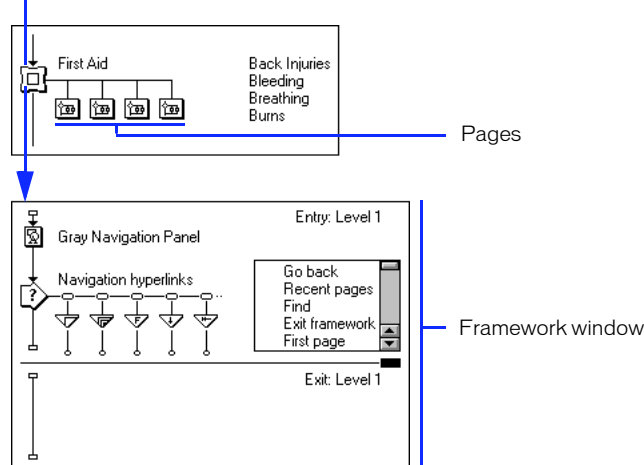
You can place a navigate icon anywhere on the flowline or use it anywhere you'd use any other icon: inside a map icon, attached to a decision or interaction icon, or attached to a framework.

A navigate icon's destination must be a page attached to a framework. You can't select an icon on the main flowline as the destination. Its destination can be a page attached to a different framework, but not attached to a framework that's in a different file. If you nest one framework inside another, you can select a page attached to the nested framework as the destination of a navigate icon; but if you attach a map icon to a framework, you can't select a specific icon inside the map as the destination. Although these may appear as limitations, they help ensure that your code is well-organized.

How the framework icon works

The framework icon sets up an environment designed to make creating navigation easy. The elements of the environment appear in the framework window.

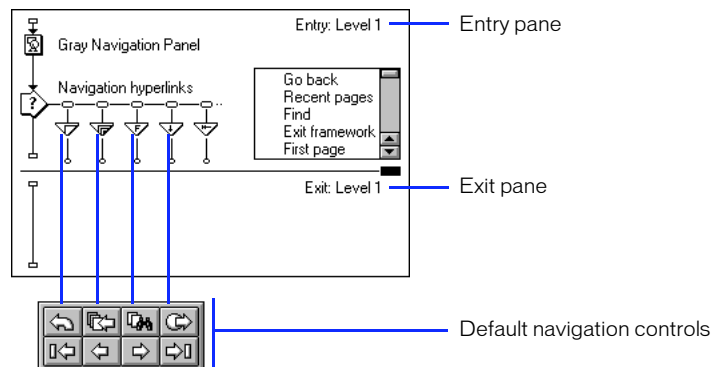
Framework icon



The environment has three components: the framework window, a set of default navigation controls, and the icons attached to the framework icon. (Any icon attached directly to a framework icon is called a page ID.)

The framework window has two parts: an entry pane and an exit pane.

The framework environment works like this: When Authorware enters the framework, it first goes to the flowline in the framework window's entry pane.



The icons in the entry pane affect every page that's attached to the framework icon. For example, if you set up a display icon in the entry pane to display a background, the background appears on every page. The default navigation controls are part of the entry pane, so they appear on every page as well.

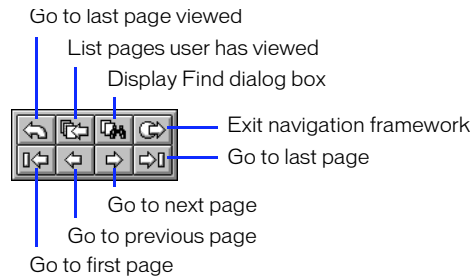
Authorware goes to the flowline inside the entry pane whenever it enters the framework—no matter to which framework page it's headed. If Authorware is coming from somewhere outside the framework directly to the third icon attached to the framework, for example, it still goes to the flowline inside the entry pane first. That's how Authorware makes sure that the environment you set up in the entry pane affects every page in the framework, no matter what order they appear.

Before Authorware exits the framework, it does two things:

- It returns to the framework window and goes to the flowline in the exit pane.
Anything you set up there happens each time Authorware leaves the framework. For example, if you want to change the value of a variable each time a user exits the framework, you'd set up a calculation icon in the exit pane to do that.
- It erases everything in the entry pane and in any page in the framework.
This includes any buttons you've set up as perpetual. In short, it returns everything to the way it was when it entered the framework.

The framework's navigation controls

The Authorware default navigation controls work well for most common navigation structures.



However, it's easy to change the controls or to replace them entirely with a set of custom controls. Just open up the framework window and edit the navigation control's interaction—or delete the interaction and create a new one.

You can even make a set of custom controls a permanent part of the framework icon. The framework icon—and the default navigation controls inside it—are contained in a model named `Framework.a7d` that's in the same folder as Authorware. If you want to make a custom set of controls a permanent part of the framework icon, replace the current model with a model that contains a framework icon with the new navigation controls inside it. You can do this easily by dragging the modified framework icon to the framework icon on the Icon palette.

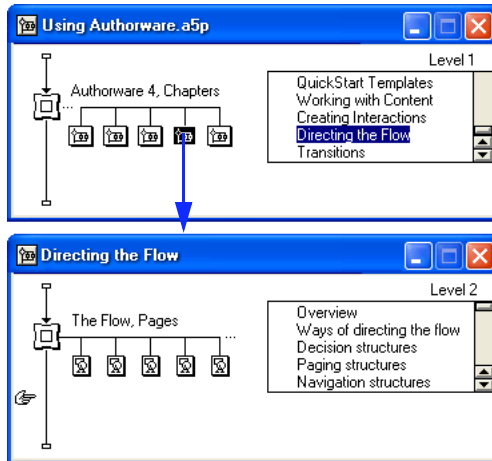
You can customize the framework icon model even further if you want. For example, if you use a layout template to lay out the screen, you can include it in the model, and it will appear on every page you attach to the framework icon.

Nesting navigation frameworks

Although any icon attached directly to a framework icon is called a page, a page doesn't have to be just a screen full of text. It can be a digital movie icon, a sound icon, or even a map icon containing another framework.

Nesting one framework inside another is a good way of structuring material and providing users with quick access to the information they're interested in.

For example, imagine an electronic version of a book. One way to set it up is to use a single framework icon with a display icon attached for every page in the book. A better way to organize the piece is to create a framework in which each page is a map icon containing an entire chapter and then to nest inside each map icon another framework in which each page is a page of text.



With the piece organized that way, users can jump directly from the beginning of one chapter to the beginning of the next. They don't have to page through all of one chapter to get to the next.

Things to consider about navigation structures

Here are some things to think about when you're designing navigation:

- Are you designing the piece to run over the Internet?
If so, remember that downloading data over the Internet takes longer than reading the same data from a local hard disk. Consider your overall navigation design carefully.
- What navigation controls do you need?
If the default navigation controls aren't what you want, change them or create a set of custom controls.
- How should the content be structured?
If the information you want to present is too deep for a single framework, consider a nested structure. Modify the navigation controls to make it easy for users to get where they want as quickly as possible.

- Are any text or graphic elements common to everything inside the navigation framework?
If so, set those elements up in the entry pane of the framework icon.
- Is there anything you want Authorware to do every time users enter the navigation framework?
For example, do you want Authorware to set the value of a variable whenever users enter the navigation framework? If so, set it up to happen in the entry pane.
- Is there anything you want Authorware to do every time users exit the navigation framework?
If so, set it up to happen in the exit pane.

Navigation structures—step-by-step procedures

What you can do	In Authorware Help, see
Set up a navigation framework	Setting up a navigation framework
Set up a framework's entry and exit panes	Setting up a framework's entry and exit panes
Import text into a paging structure from an RTF file	Importing text into a framework
Reposition the default navigation controls	Repositioning the navigation controls
Select a transition effect between pages	Setting up page transitions
Disable Previous and Next buttons so piece doesn't go from last page to the first or from first to last	Preventing wraparound paging Paging structures—step-by-step procedures
Change the way a navigation control works	Changing where a navigation control takes a user
Change the look of a navigation control or the sound it makes	Changing a navigation control's graphic or sound
Provide a key that a user can press instead of clicking a navigation control	Assigning a shortcut key to a button
Make a navigation control active only under certain conditions	Limiting when a button is active
Hide or dim a navigation control when it's not active	Hiding or dimming an inactive button
Make a navigation control the default	Making one button the default
Add or delete a default navigation control	Adding or deleting a default navigation control
Create custom navigation controls	Creating custom navigation controls
Save navigation controls as a model	Saving navigation controls as a model
Replace the default navigation controls with a model	Replacing the default navigation controls with a model

Navigation links—jumping from place to place

To create a navigation link in Authorware, use a navigate icon.

Remember, you can place a navigate icon anywhere on the flowline or use it anywhere you'd use any other icon—inside a map icon, attached to a decision or interaction icon, or inside a framework—but its destination must be attached to a framework.

Anywhere—going to a specific page

The simplest and most straightforward type of navigation link is a link connected with a specific page. Unlike the Next and Previous links, whose destinations change each time Authorware moves to a different page, the destination of a navigate icon linked to a specific page never varies.

You'll use Anywhere links most often for things such as menus—where you know you want a link to go to exactly the same place each time.

Recent—backtracking

Many times it's important to give users an easy way to get back to places they've been before. Perhaps you're creating an electronic travel guide. After paging through descriptions of dozens of hotels, a user is likely to want to return to one or two and find out more about them. Another user who's searching for something specific in the guide—jumping around, rather than going from one page to the next—is occasionally going to want to return to the last entry before jumping forward to the next.

The Recent navigation link makes two sorts of backtracking possible:

- Go Back takes a user back to the last page Authorware displayed. It has nothing to do with physical proximity—that's handled by the Previous and Next navigation links. Go Back goes to the page the user last visited, regardless of where it is.
- List Recent Pages displays a list of the pages where Authorware has been most recently. The most recent page is at the top of the list. The user can select any page from the list and return to it. Authorware displays the icon title of each page, so it's a good idea to give icons clear, descriptive names.

It's important not to confuse backtracking with a call (see [“One-way trips and round trips” on page 146](#)). A call is an automatic round trip: Authorware goes to a destination, and when it's finished there, it snaps back to its point of departure as if it were at the end of a tightly stretched rubber band. Backtracking isn't automatic. It allows users to select to return to places they've already been.

Nearby—paging

The Nearby options in a Navigation icons properties are best suited for setting up navigation links between pages in a navigation framework. They're designed to automatically keep track of where you are and to take you to the places where you're most likely to want to go—to the next or previous page or to the first or last page.

One of the Nearby options—Exit Framework/Return—also takes care of the tricky task of getting you back to where you started when you jump from one framework to another.

Calculate—using an expression to go to a page

Sometimes it's convenient to use a variable or expression to define the destination of a navigation link rather than selecting a specific icon as the link's destination.

For example, suppose you've created a piece on first aid that includes a glossary of terms. The word *angina* might appear in a dozen different places in the text. One way to link each instance of *angina* with its definition in the glossary is to manually set up a separate link for each instance of the word. That's not very efficient. You'd have to repeat the same process not only for every instance of the word *angina*, but for every instance of every word that appears in the glossary.

A more efficient way to approach the task is first to assign the keyword *angina* to the page that contains the definition of *angina*. When a user clicks *angina* to find out what it means, Authorware automatically saves the word the user has clicked in a system variable. You can then use the variable to search for the keyword and find the page you've assigned it to. This approach is especially efficient because you need to set up the expression only once. You can actually make the expression part of the definition of the hot-text style you apply to glossary terms, and you can use the same expression to find any word in the glossary.

Search—allowing users to search for a page

The Search navigation link lets users search for a word and go to any of the pages where the word appears. The Search navigation link is typically used for a Find button, like the one on the Authorware default navigation controls panel.



When a user clicks the Find button, the Find dialog box appears.

You can set up the Find dialog box to search either an entire file or just the current framework. You can also search not only for the text in text objects that are inside display and interaction icons but also for the keywords you've assigned to icons. Searching for keywords makes it possible for users to locate icons that don't display text, such as digital movie or sound icons.

Users can also set up sophisticated searches based on Boolean logic. For example, they can set up a search to find:

- All the pages that contain two or more words, either in any order (Cairo & Paris & Moscow) or in a particular sequence (Cairo, Paris, Moscow)
- All the pages that contain any of several words (Cairo | Paris | Moscow)
- All the pages that contain certain words but not others (Cairo & Paris ! Moscow)

You can preset the search text in the Find dialog box with either literal text or a variable. Presetting the search text gives you a way to automate the search process for users. For example, Authorware has a system variable called `WordClicked`, in which it stores any word the user clicks. By using `WordClicked` as the preset text and selecting the Search Immediately option, you can allow users to start a search by clicking any word on the screen.

You can also set up the Find dialog box to show not just the name of the page where the search text has been found but also part of the line where the word appears.

The context can help users identify which page they want to go to.

One-way trips and round trips

Several of the navigation links give you two ways to travel. You can take users either on a one-way trip, known as a jump, or on a round trip, known as a call.

A jump is a simple, straightforward trip from a current location to a destination. A call keeps track of the current location so that after it arrives at its destination, it can return to the spot it came from.

Let's say you've built a help system into the course you've created. You want to structure the course's navigation so that users can get to the help system from anywhere in the course and return automatically to the place they came from when they're finished. To do that, you'd call the help system rather than jump to it.

The navigate icon where a call starts (and to which it eventually returns) can be either inside a framework or on the flowline outside. However, the destination of the outbound leg of the round trip must be a page attached to a framework icon. There's also one restriction on using a call that doesn't apply to a jump. You can jump between pages within the same framework, but you can call a page only if it's in a different framework.

A call requires two navigate icons: one to take Authorware to the destination and another at the destination to take Authorware back where it came from. (Either navigate icon can also be replaced with a hot-text hypertext link.) At the destination, the navigate icon that sends Authorware on its return trip must be set to Exit Framework/Return.

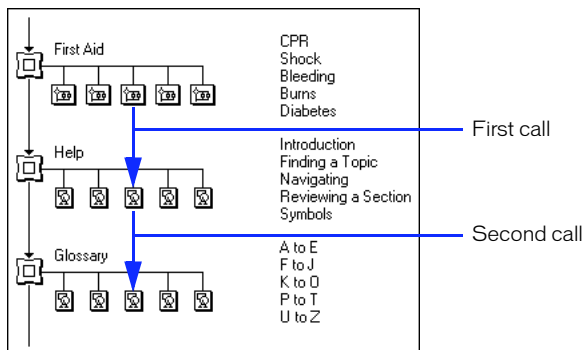
When Authorware returns, it doesn't go back to the exact place it came from. If it did, it would return to the navigate icon where it started its journey and start the trip all over again. Instead it goes to the point on the flowline immediately following the place it departed from. If you attach a map icon to a framework and then set up a navigate icon inside it to call another page, when Authorware returns to the map, it returns not to the beginning of the map, but to the place in the map immediately following where it left from.

All the different types of links allow you to set up a jump. The following three also let you set up a call:

- Anywhere links to a specific page.
- Calculate links to a page that's identified by an expression you've created.
- Search allows users to search for the page where they want to go.

Nesting calls

When you set up a call, you don't have to return immediately. For example, you can call one page, and from the first page call a second, and from the second page call a third. That's known as nesting calls.



When you nest calls, it's important to remember that when it's time to return, the navigate icon that sends Authorware back has no idea where Authorware came from. It just tells Authorware to return and assumes that Authorware knows where to go. Authorware does. It keeps track of where it came from by keeping a list. Every time you call a page, Authorware records its current location on the list and then goes to the new destination. When a navigate icon tells it to return, it goes back to the last place on its list, then deletes that location from the list.

Don't confuse Authorware by trying to use a navigate icon linked to a specific destination to send it back to its previous location. If you do, Authorware won't look at its list, and it won't delete the location at the bottom. The next time a navigate icon tells it to return, it will check the location at the bottom of the list—and end up someplace you don't expect.

Navigation links—step-by-step procedures

What you can do	In Authorware Help, see
Set up a navigation link to a page attached to a framework icon	Setting up a navigation link
Go to the next, previous, first, or last page	Going to the next, previous, first, or last page
Go directly to a specific page	Going to a specific page
Go to a page, then return to the place where the user started	Going to a page and returning
Use the value of a variable or an expression to determine the page to go to	Using an expression to go to a page
Let users search for a page	Allowing users to search for a page
Let users search for a topic discussed on a page instead of the text that's on the page	Allowing users to search for a topic
Assign a topic to a page	Assigning a keyword to a page
Copy topics from one page to another	Copying keywords from one icon to another

What you can do	In Authorware Help, see
Change a topic	Editing a keyword
Delete a topic	Removing a keyword
Customize the Find dialog box	Setting up the Find dialog box
Let users go back to pages they've already visited	Allowing users to backtrack
Customize the dialog box that displays the list of recent pages	Setting up the Recent Pages dialog box
Nest one navigation framework inside another	Nesting navigation frameworks

Navigating with hypertext

When you attach a navigation link to hot text—text that users can click like a button—you're creating hypertext.

Creating hypertext links in Authorware Authorware has a uniquely easy method for creating hypertext links. You first create a text style with interactivity built into it. You then apply the style to any text you want to turn into a hypertext link and select the destination Authorware goes to when a user clicks the text.

The hypertext style is really a style with a navigate icon embedded in it. Setting up hypertext navigation is just like setting up a navigation link once you have the style defined.

Hypertext—step-by-step procedures

What you can do	In Authorware Help, see
Create hypertext	Navigating with hot text
Create a hot text style	Creating a hot text style
Apply the hot text style to text	Applying a hot text style

Jumping to another file

Often your project is big enough that you need to create several Authorware files to encompass the whole thing. If you're creating an electronic how-to guide, for example, you may want to make each chapter, along with its graphics, animations, and digital movies, a separate file.

As mentioned earlier, directing the flow from one file to another is different from directing the flow from one icon to another within a single file. You can't use a navigate icon to jump from one file to another. Instead you need to use one of two system functions: `JumpFile` or `JumpFileReturn` (see *System functions in Authorware Help*). `JumpFileReturn` allows Authorware to return to the file it came from automatically; `JumpFile` doesn't.

You can attach the `JumpFile` and `JumpFileReturn` functions to a navigation control just as easily as you attach a navigate icon. In effect, the two functions work so much like a navigate icon that users may never know whether they're jumping from one place in a file to another or from one file to another.

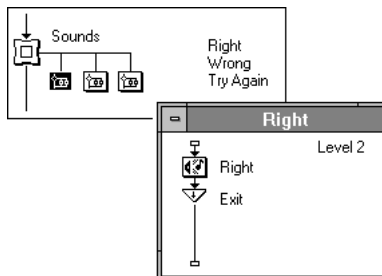
If you're designing an Authorware piece to run over the Internet, keep the following things in mind when you're setting up jumps from one file to another:

- Think small—avoid long downloads. Internet access is slower and takes longer than local access. Make each file as efficient as possible.
- Orchestrate downloads. Because it takes longer to download a file over the Internet than to read the same file from a local hard disk, download files when it's least noticeable. Don't make users wait. Otherwise the piece will seem slow and unresponsive.
- Think about security. Authorware lets users select to trust or not trust sites they're downloading from; it displays a dialog box at the beginning of each file it downloads to let them make the choice. If you avoid using certain functions and variables that have the potential to violate the security of a user's system, you can turn the Security dialog box off.

Subroutines

If you find yourself using the same sequence of icons over and over again in the same piece, consider setting them up as a subroutine. A subroutine is any single sequence of icons you've set up so that Authorware can use it in many different places. Online help systems and glossaries are examples of large subroutines. The essential characteristic of a subroutine is that Authorware can go to it and then return to the exact spot it came from.

For example, suppose you want to use the same three sounds over and over in a piece. You could set up a separate sound icon for each sound each time you wanted to use it. A more efficient approach is to attach the sounds to a framework icon at the end of the piece, like this:



Each page in the framework is a map icon that contains one of the sounds and a navigate icon that's set to exit the framework. Wherever you want to use one of the sounds, you'd set up a navigation link—a navigate icon or hot text with the Call and Return option set—to go to the page in the framework that contains the appropriate sound icon.

The subroutine works like this: When Authorware encounters one of the navigation links you've set up, it goes to the subroutine, plays the sound, and then continues down the flowline to the navigate icon. The navigate icon directs the flow to exit the framework. Because you set the Call and Return option for the navigation link that directed Authorware to the sound, Authorware exits the framework and automatically returns to where it came from.

The basics for creating a subroutine are simple. The subroutine must be attached to a framework icon. The navigation link that exits the subroutine must be set to Exit Framework/Return. The navigation link that directs Authorware to the subroutine must have the Call and Return option selected.

CHAPTER 7

Transitions, Positioning, and Motion

Transitions and motion create visual interest in a piece. Transitions are special effects that occur when Macromedia Authorware displays or erases objects on the screen. Television and movies often use transitions between scenes—scenes softly fade away or dramatically spiral from one to the next.

Objects that move can draw attention to content; they can be part of an interaction, or they can simulate the behavior of real-world objects. Keep in mind that transitions and moving objects alone won't make a piece interesting. Interactions and high-quality content should take priority over motion and special effects.

Positioning has to do with changing an object's position in response to conditions that you set. Positioning is a helpful way to give visual feedback to users. For example, as a student finishes certain sections of an educational piece, a marker might move along a progress bar or a map of the course.

This chapter describes the following topics:

- Using transitions to achieve special effects
- Positioning objects
- Moving objects on the screen
- Letting users move objects by dragging

Using transitions for special effects

Transitions in Authorware are special effects applied to the Presentation window as you either display or erase objects. For example, the Push transition might slide a new screen over an old one; the Dissolve transition might slowly reveal a test answer or a graphic.

Skillful use of transitions can dramatize information, give greater impact, and enhance the overall aesthetics. Transitions can also help the rhythm and pacing of the piece and orient a user. For example, you can use a subtle transition when moving from page to page and a dramatic transition when moving from section to section.

Transition settings

You select transitions and the settings associated with them in the Transitions dialog box (select Modify > Icon > Transition).

You can vary the duration and smoothness of a transition and select whether a transition affects the entire Presentation window or just the area in the window where new objects appear. You can use a variable or an expression to set duration and smoothness. For example, you can create a custom variable for the duration of a transition used frequently in a piece and assign its value in a calculation icon at the beginning. Later, if you decide the transition should run slightly faster or slower throughout the piece, just change the value of the custom variable.

Some transitions have other options corresponding to the effects they create. For example, The Byte (SharkByte Transitions) lets you control such elements as the number of teeth, their length and jaggedness, and whether a sound plays with the transition.

Transition Xtra extensions

Transition Xtra extensions supplement Authorware built-in transitions and are listed along with the built-in transitions in the Transitions dialog box. The Director Transitions Xtra—which provides the complete set of more than 50 transitions from Macromedia Director—as well as transitions from g/matter, Xaos Tools, and Zeus Productions are installed with Authorware.

The total number of transitions available for use with a piece depends on the transition Xtra extensions available in the Authorware Xtras folder. In addition to the transition Xtra extensions installed with Authorware, you can install transition Xtra extensions from independent developers.

When you distribute a piece, remember to provide users with the Xtra extensions that you've used. These Xtra extensions must be installed in the Xtras folder within the folder that contains the Authorware run-time application. See [“Determining which files you need to distribute” on page 56](#).

Positioning objects

Positioning an object is different from moving an object. Moving an object means changing an object's position on the screen as a user watches. Positioning an object means defining where it appears on the screen. An object may appear in different places at different times or for different users. But it simply appears at the position you designate; it doesn't move there.

Objects normally appear exactly where you position them in the Presentation window. However, sometimes you might want the position of an object to depend on an event or a changing condition in the piece—for example, a value users enter, the amount of time that has passed, or the position of other objects. At other times you might want users to be able to drag objects wherever they want.

You can position text, graphics, sprites, buttons, or digital movies at varying points in several different ways:

- By positioning an object anywhere on the screen (see Positioning objects on the screen in Authorware Help)
- By positioning an object on a path (see Positioning objects on a path in Authorware Help)
- By positioning an object within an area (see Positioning objects within an area in Authorware Help)
- By letting users drag objects wherever they want (see Dragging an object in Authorware Help)

Positioning objects on a path

You might want to position an object at different points along a path. A path can contain straight or curved line segments. You can specify the path and the base and end values for the path's ends, and then insert a variable or expression that determines the object's position.

For example, if you want to show users their progress through a series of training courses, create a screen that lists the various sections in the series along a line. As users complete each section, set the value of a variable to correspond to the section's position on the line. At the beginning and end of each section, use the variable to position a marker on the line—and show users how far they've gone.

Use the On Path display icon property to position an object at different points along a path—for example, to create a gauge or slider that moves to a specific point on the path depending on the value of a variable.

Positioning objects within an area

You might want to position an object at various points within a specific area. To do so, specify the area, assign horizontal and vertical values to the area's corners, and define variables or expressions that set the object's horizontal and vertical position.

For example, suppose you're creating a series of lessons on chess strategies. At the beginning of each lesson, you want to display a chessboard with the pieces set up to illustrate a particular strategy. To do that, position the pieces by setting the value of the variables assigned to them. Once you have the pieces positioned, use motion icons to move the pieces to their final positions to demonstrate the strategy that the lesson is about.

The In Area display icon property lets you specify the area, assign horizontal and vertical values to the area's corners, and define variables or expressions that set the object's horizontal (X) and vertical (Y) position.

Positioning objects on the screen

You can position an object anywhere on (or off) the screen using both X and Y coordinates. Either use numeric values for the coordinates to place the object in a set location, or use variables for the coordinates to place the object in varying locations depending on the value of the variables. For example, if you want to let users see the definition of a term by clicking it, use the `ClickX` and `ClickY` variables to display a pop-up window containing the definition at the location where the user clicked.

The On Screen display icon property places objects anywhere on the screen.

Making objects move

Use the motion icon to move objects around the screen. The motion icon moves objects either directly to a point (on a line, a grid, or anywhere in the presentation window) or along a straight or curved path.

You can vary the speed or distance an object moves along a path in response to certain conditions in the piece. For example, you could simulate a steel ball rolling within a magnetic field or a ball thrown straight up into the air. The objects could move varying distances, depending on the strength of the magnetic or gravitational field.

You can also set the scale of grids and paths by using constants, variables, or expressions. The scale is 0 to 100 by default, but you might want to change the scale, say for a timeline or for a Kelvin-scale thermometer.

You can move more than one object at a time. For example, in a demonstration of the solar system each planet could move independently, controlled by a different motion icon.

Using the motion icon

Motion has two components—the motion icon and the icon containing the object you want to move. The motion icon moves everything that's contained in the icon you select—text as well as graphics. Place objects that you don't want moved, such as backgrounds and text, in separate icons.

The motion icon gives you a choice of five motion types:

- Direct to Point
- Direct to Line
- Direct to Grid
- Path to End
- Path to Point

Moving objects directly to a point Direct to Point moves an object to the same final destination no matter where it currently is on the screen. For example, you might want to show the correct place for an object after you've allowed the user to move it.

The Direct to Point motion icon moves objects from *any* position on the screen to the location you select.

Moving objects to a point on a line Direct to Line motion moves an object from anywhere in the Presentation window to a specific point on a line. When the object is already on the line, it appears to slide along the line; when the object isn't on the line, it moves toward the specified point. For example, if you want to show the relationship of historical events to each other on a timeline, you can move a marker from a description of the event to its proper place on the timeline.

You can tailor the range and scale of the line to your needs.

Moving objects to a point on a grid Sometimes, rather than moving an object to a point on a line, you may want to move an object to a point on a grid. For example, you might want to move pieces vertically and horizontally on a chess board.

The Direct to Grid option moves an object to specific horizontal and vertical coordinates on a grid that you define.

Moving objects along an entire path In some situations you'll always want an object to move along the same path from beginning to end. For example, you might want to show the orbit of a planet or a bird's flight path.

The Path to End motion icon moves an object from beginning to end along a path that you define.

Moving objects to a point on a path You may want to move an object along a path to some point that varies depending on some condition that you're tracking—for example, if you're creating a thermometer or a gauge in a simulation.

The Path to Point motion icon moves an object along a path until it reaches some specific value. To track a condition that may change while the piece is running, set the value with a variable.

Letting users drag objects

You can let users move objects by dragging them around the Presentation window, within a specific area, or along a specific path. Dragging objects is a common form of interactivity. You can let users assemble equipment, match two items by dragging one to the other, control some process in a simulation by dragging a slider, and so on.

Making an object movable Authorware freezes everything in place when you package a piece so users don't accidentally move things around the screen. If you want to let users move an object, you must make the object movable, either with the Movable setting for the icon that contains the object (Modify > Icon > Properties) or with the variable `Movable`. Using the `Movable` variable is more flexible than using the Movable setting because you can turn the variable on and off to control precisely when users are able to move an object.

Allowing users to interact There are two ways to let users move an object once you've made it movable. One is to set up a target area interaction. Target area interactions are ideal if there's some specific spot you want users to drag an object to—for example, if you're letting users assemble equipment or match two items by dragging one to the other. But they're not so effective if you're, say, setting up a slider to control a process in a simulation. The slider can be anywhere along the scale—there's no one position that's its target.

The second way to let users to move an object works for things such as sliders. It involves using the Positioning setting for the icon that contains the object (Modify > Icon > Properties) to define where the object can be moved and using the variable.

PathPosition tracks the location of the object. For an example and a description of how the approach works, see the Show Me: “[Slider \(Slider.a7p\)](#)” on page 229.

Variables for tracking an object's motion and position

Authorware has a set of variables designed specifically for tracking an object's motion and position. The variables are listed in the following table along with a brief description. For more information about them, see Chapter 15, System Variables in Authorware Help.

Variable	Description
<code>Animating</code>	Indicates whether an object is being moved by a motion icon.
<code>DisplayHeight</code>	Contains the height of an object.
<code>DisplayLeft</code>	Contains the distance between the left end of an object and the left edge of the Presentation window.

Variable	Description
DisplayTop	Contains the distance between the top of the object and the top edge of the Presentation window.
DisplayWidth	Contains the width of an object.
DisplayX	Contains the distance from the center of the object to the left edge of the Presentation window.
DisplayY	Contains the distance from the center of the object to the top of the Presentation window.
Dragging	Indicates whether a user is dragging an object.
Movable*	Indicates whether the user can move an object.
Moving	Indicates whether an object is being moved by a motion icon or dragged by a user.
PathPosition	Contains the position of the object on a path.
PositionX PositionY	Contains the vertical and horizontal position of an object in an area.

Movable can be set by the developer.

Making objects move—step-by-step procedures

The following table lists the tasks you can accomplish and where you'll find the steps to do so.

What you can do	In Authorware Help, see
Move an object from anywhere to a specific position	Moving objects direct to a point
Move an object from anywhere to a specific point on a line	Moving objects to a point on a line
Set the base and end points of a line	Moving objects to a point on a line
Move an object from anywhere to a specific point on a grid	Moving objects to a point on a grid
Create a grid	Moving objects to a point on a grid
Move an object along a path from beginning to end	Moving objects along an entire path
Move an object along a path to a specific point	Moving objects to a point on a path
Create a path	Creating paths, Making circular paths

CHAPTER 8

Using Variables, Functions, Xtra Extensions, and More

The Macromedia Authorware variables and functions give you a level of control that's much greater than what you can achieve by manipulating icons on the flowline. You can collect and manipulate data, get detailed information about interactions, make decisions based on specific conditions, and set up Authorware to do just about anything a computer is capable of.

You don't need to be a programmer to take advantage of variables and functions—in many cases, Authorware makes it as easy as copying and pasting.

Variables represent values that can change. As the word implies, a variable's value varies. Authorware contains over 250 variables that are ready to use. For example, the system variable *PercentCorrect* automatically keeps track of the percentage of questions a user has answered correctly. You can easily create your own custom variables to store any information you need to keep track of.

Functions perform specific tasks. Authorware has more than 350 functions. For example, the system function *JumpFile* lets you jump from one Authorware file to another. *WriteExtFile* lets you save information in a text file.

You can combine variables and functions in expressions. An expression is any part of an Authorware statement, meant to be taken as a whole. For example, $2 + 2$ is an expression but not a valid statement all by itself. A statement is a valid Authorware instruction that produces a result by performing a calculation or carrying out an operation. For example, `UserName := "John Smith"` is a statement that assigns a value to a variable.

To add to your expressions, you can insert commonly used programming statements, called code snippets, such as if...then statements and repeat loops, into your calculation. You can select from default snippets, edit them, or add your own.

Combining variables, functions, and expressions—along with code snippets—to tell Authorware exactly what you want it to do is called scripting.

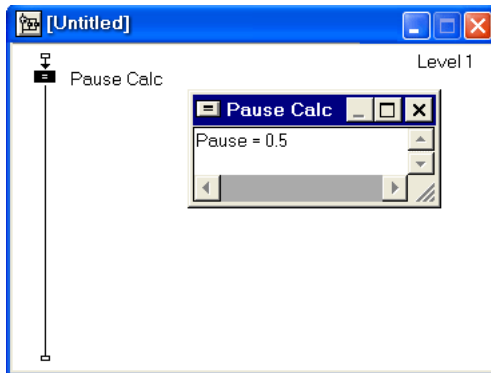
What each variable and function does

Each variable and function has a specific purpose. To use a variable or function properly, you need to know exactly what it's designed to do. There are three places you can find out:

- “System Variables” and “System Functions” in Authorware Help describe each variable and function in detail.
- You can download PDF versions of System variables overview and System functions overview from www.macromedia.com/support/authorware/.
- The Variables dialog box gives a brief description of each variable. The Functions dialog box lists the syntax and gives a brief description of each function.

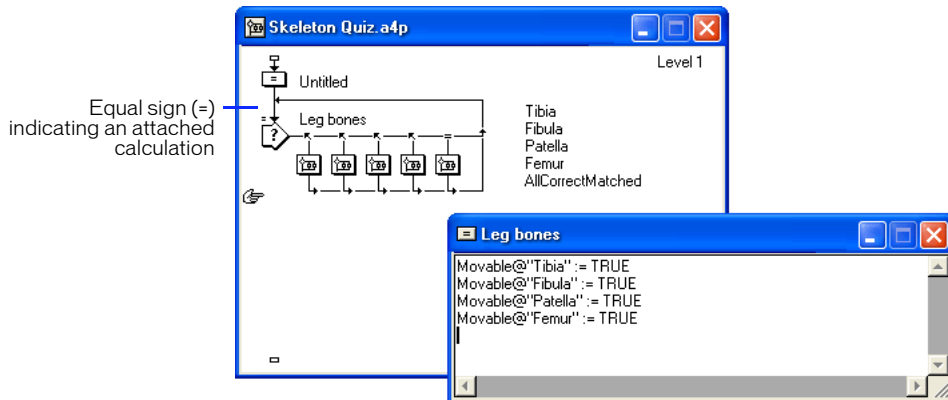
Where you can use variables, functions, expressions, code snippets, and scripts

In a calculation icon Opening a calculation icon displays a Calculation window. Calculation windows can contain variables, functions, expressions, code snippets, and scripts. For example, you might put a calculation icon at the very beginning of a file and use it to set the value of a variable that controls how long wait icons pause throughout the file.



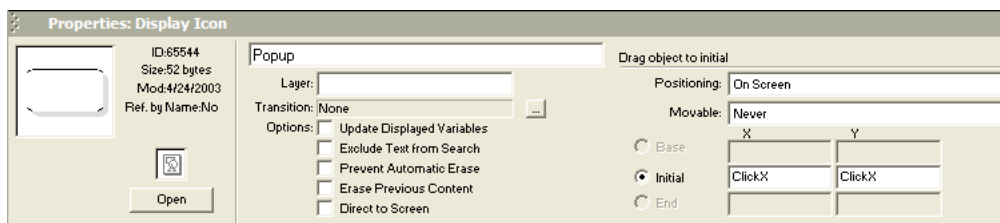
In a Calculation window attached to an icon You can also use variables, functions, expressions, code snippets, and scripts in Calculation windows attached directly to icons.

You attach a Calculation window directly to an icon to make sure Authorware executes the calculation whenever it encounters the icon. For example, you've created a target area interaction and you want to make sure that the objects users are supposed to move are tagged as movable at the moment they enter the interaction. The best way to accomplish this is to attach a Calculation window to the interaction icon (Modify > Icon > Calculation) and set it up as shown in the following figure:



When you attach a Calculation window to an icon, a small equal sign (=) appears slightly above and to the left of the icon.

In the Property inspector You can use variables, functions, and expressions—but not scripts—in most editable fields in the Property inspector. For example, you might want to position an object at the last spot where the user clicked. To do that, use the *ClickX* and *ClickY* variables in the Positioning fields of the Property inspector, as shown in the following figure:



In a few editable fields, you can't use a variable, function, or expression: fields where you name things (such as icons, text styles, or keywords) or where you provide descriptions (such as the description of a button state in the Button Editor). There are also other fields where only literal values can be used, such as the scale and size fields of image properties.

Embedded in display text You can display the contents of a variable or the value of an expression on the screen by embedding it in a text object or an RTF document. Embedding the content of a variable in text objects makes it possible to display information that changes from moment to moment or from user to user. Embedding an expression in a text object makes it possible to display the result of a calculation without having to add a calculation icon to the flowline. A text object with embedded variables and expressions looks like this:

Your Score	
Part 1	{Score 1}
Part 2	{Score 2}
Part 3	{Score 3}
Total	{{(Score 1+Score 2+Score 3)/3}}

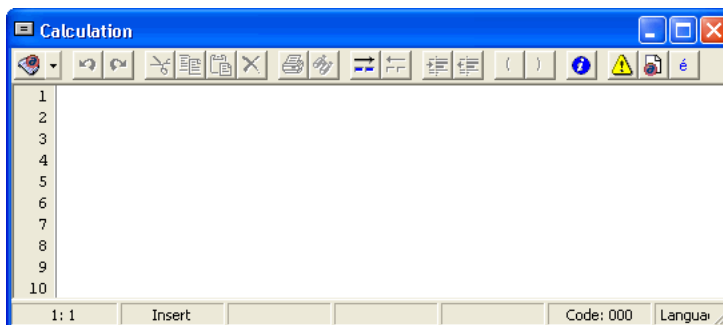
When Authorware displays the text object, it displays the value of the variable or expression, not the variable or expression itself.

Your Score	
Part 1	87
Part 2	91
Part 3	86
Total	88

Using the Calculation window

You can use the Calculation window to do the following tasks:

- Paste in a function or variable, or insert a code snippet.
- View the ASCII value of the selected character in the Calculation window. [“Displaying the ASCII value of selected text” on page 161.](#)
- Change calculation preferences for defaults in the Calculation window. For example, you can set up your calculation preferences so that case-sensitive functions or variables are automatically converted to the correct case. For more information, see [“Changing calculation defaults in the Calculation window” on page 161.](#)
- Insert a symbol, divider line, or message box into a calculation. See [“Inserting a message box, snippet, or symbol into a calculation” on page 161.](#)

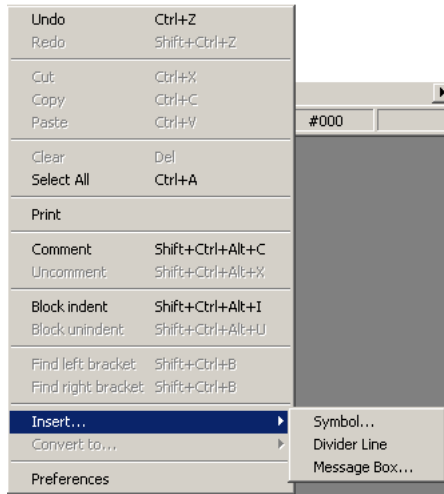


Changing calculation defaults in the Calculation window

You can change several preferences in the Calculation window. After you change the preferences, they become the defaults. For example, you can set up your calculation preferences so that case-sensitive functions or variables are automatically converted to the correct case.

Inserting a message box, snippet, or symbol into a calculation

You can access the context menu in the Calculation window by right-clicking from the Calculation window.



You can select any toolbar option from this menu as well as insert a message box, snippet, or symbol into the calculation.

Displaying the ASCII value of selected text

You can view the ASCII value of selected text in the Status bar in the Calculation window.

Calculation window keyboard shortcuts

You can use the following keyboard shortcuts in the Calculation window.

Keyboard Shortcut	Result
Up arrow	Move up one row
Shift+Up arrow	Select the row above
Down arrow	Move down one row
Shift+Down	Select the row below
Left arrow	Move one character to the left
Shift+Left arrow	Select characters to the left of the insertion point
Control+Left arrow	Move to the word to the left

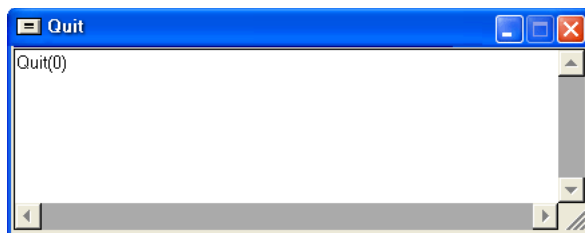
Keyboard Shortcut	Result
Shift+Control+Left arrow	Select the word to the left
Right arrow	Move the insertion point one character to the right
Shift+Right arrow	Select characters to the right of the insertion point
Control+Right arrow	Move to the word to the right
Shift+Control+Right arrow	Select the word to the right
PgDn	Move one page down
Shift+PgDn	Select from the insertion point to the bottom of the page
Control+PgDn	Move to the bottom of the page
Control+Shift+PgDn	Select from the insertion point to the bottom of the page
PgUp	Move one page up
Shift+PgUp	Select from the insertion point to the top of the page
Control+PgUp	Move to the top of the page
Control+Shift+PgUp	Select from the insertion point to the top of the page
Home	Move to the beginning of the line
Shift+Home	Select from the insertion point to the beginning of the line
Control+Home	Move to the beginning of the editor
Shift+Control+Home	Select from the insertion point to the top of the Calculation window
End	Move to the end of the line
Shift+End	Select from the insertion point to the end of the line
Control+End	Move to the end of the editor
Control+Alt+[X]	Move to line [X]
Shift+Control+End	Select from the insertion point to the bottom of the Calculation window
Alt+Shift	Select column mode
Alt+Shift+Up	Select column up
Alt+Shift+Down	Select column down
Alt+Shift+Left	Select column left
Alt+Shift+Right	Select column right
Alt+Shift+Home	Select column to the beginning of the line
Alt+Shift+End	Select column to the end of the line
Alt+Shift+PgUp	Select column to top of page
Alt+Shift+PgDown	Select column to bottom of page
Control+Alt+Shift+Home	Select column to the top of the screen
Control+Alt+Shift+End	Select column to the bottom of the screen

Keyboard Shortcut	Result
Control+Alt+Shift+PgUp	Select column to the beginning of the editor
Control+Alt+Shift+PgDn	Select column to the end of the editor
Control+Alt+Shift+Left	Select column to the word to the left
Control+Alt+Shift+Right	Select column to the word to the right
Ins	Toggle between insert and overwrite mode
Delete	Delete character at right
BkSp	Delete character at left
Control+BkSp	Delete word to left
Enter	Line break
Control+A	Select all
Control+C	Copy
Control+Insert	Copy
Control+V	Paste
Shift+Insert	Paste
Control+X	Cut
Shift+Delete	Cut
Control+Z	Undo
Alt+Backspace	Undo
Shift+Control+Z	Redo
Shift+Alt+Backspace	Redo
Escape	Close calculation icon
Tab	Tab forward four characters
Shift+Tab	Move backward four characters
Shift+Control+B	Find matching bracket in an expression
Numeric keypad Enter key	Closes and saves the contents of the Calculation window.
Control-Alt-Enter	Closes and saves the contents of the Calculation window (on laptop keyboards)
Alt+Enter	Create a continuation character
Control+Shift+123	Go to line number 123
Control+Shift+Alt+I	Block indent
Control+Shift+Alt+U	Block unindent
Control+Alt+C	Mark line as a comment
Control+Alt+X	Unmark line as a comment

Keyboard Shortcut	Result
Shift+Control+0	Toggle bookmark 0
Shift+Control+1	Toggle bookmark 1
Shift+Control+2	Toggle bookmark 2
Shift+Control+3	Toggle bookmark 3
Shift+Control+4	Toggle bookmark 4
Shift+Control+5	Toggle bookmark 5
Shift+Control+6	Toggle bookmark 6
Shift+Control+7	Toggle bookmark 7
Shift+Control+8	Toggle bookmark 8
Shift+Control+9	Toggle bookmark 9
Shift+Control+Alt+0	Go to bookmark 0
Shift+Control+Alt+1	Go to bookmark 1
Shift+Control+Alt+2	Go to bookmark 2
Shift+Control+Alt+3	Go to bookmark 3
Shift+Control+Alt+4	Go to bookmark 4
Shift+Control+Alt+5	Go to bookmark 5
Shift+Control+Alt+6	Go to bookmark 6
Shift+Control+Alt+7	Go to bookmark 7
Shift+Control+Alt+8	Go to bookmark 8
Shift+Control+Alt+9	Go to bookmark 9
Control +H	Calculation complete
Control+Shift+H	Set calculation parameters
Control+Y	Insert code snippets

Creating a Quit button

To create a Quit button, for example, you attach a calculation icon to an interaction icon, select the button response type, then set up the `Quit` function in the Calculation window, like this:



Variables and Functions dialog boxes

You can also browse the Functions and Variables dialog boxes. Select Window > Variables or Window > Functions, then select the category and the variable or function you're interested in. Information about the variable or function appears in the Description text box.

Using comments

Commenting your work is always good authoring practice. It's especially important when you're dealing with variables, functions, expressions, and scripts. When you go back to your work after a week or two to modify or debug it, you may find that the purpose of a variable or an expression isn't immediately apparent. If you comment your work thoroughly as you create it, you won't have to waste time deciphering it when you want to make changes later.

You can use comments anywhere in a Calculation window. To add a comment, type two hyphens (--) and then the text of the comment. For example:

```
total := score1 + bonus --calculates raw score
```

Authorware ignores everything after the two hyphens when it evaluates the expression.

Variable and function basics—step-by-step procedures

What you can do	In Authorware Help, see
Use a variable	Using a variable
Display the content of a variable on the screen	Displaying variable information onscreen
Create an expression	Expressions and operators
Display the value of an expression on the screen	Displaying expression values on the screen
Use a system function	System functions
Use a custom function	Custom functions in DLLs
Use the functions in a scripting Xtra extension	Scripting Xtra extensions
Attach a Calculation window to an icon	Calculation Icon

Using system and custom variables

A variable represents a value that can change—such as the amount of time a user has spent in an interaction, the percentage of questions in a test that a student has answered correctly, or whether a digital movie is currently playing. Think of a variable as the address on a mailbox: The address (variable name) is always the same, but the contents (variable value) can change under certain conditions.

There are two types of variables: system variables, which are built into Authorware, and custom variables, which you create yourself.

System variables

The Authorware built-in variables, called system variables, track a wide range of information and are grouped into 11 categories: CMI, Decision, File, Framework, General, Graphics, Icons, Interaction, Network, Time, and Video. Authorware updates the information in the system variables automatically.

System variable names in Authorware always begin with an uppercase letter and consist of one or more words with no spaces between them—for example, *Tries* and *VideoDone*. Some system variable names can be followed by the “at” symbol (@) and the title of an icon. They’re called icon variables, and you can use them to find out a variable’s value at a specific icon within a file.

Some variables complement each other, such as ClickX and ClickY. Some variables work in tandem with functions, such as the variable *FileLocation* and the function *ReadExtFile*. You can assign values to some system variables to provide Authorware with information.

For example, you can make the variable *Movable* TRUE or FALSE to tell Authorware whether or not you want users to be able to move a particular object. The values of other system variables, such as *TimeRemaining*, are assigned to them by Authorware, and you can only extract the information they contain.

Before you start working with system variables, decide what you want to do with the variable. Do you want to provide Authorware with information (for example, by setting *Movable* to TRUE to tell Authorware to allow a user to move an object)? Or do you want to use the variable to extract from Authorware information you need (for example, information that a user has entered and that Authorware has stored in *EntryText*)? Once you’ve decided what you want to do with a variable, finding it and putting it to use is easy.

Custom variables

If you want to keep track of information that Authorware isn’t recording in a system variable, you can create a custom variable. Custom variables reflect the value you assign to them. The value changes only when you explicitly change it. For instance, you might keep track of how many times Authorware has passed through a decision structure with a variable named *Counter*. You’d use the expression *Counter := Counter + 1* to add 1 to the value of *Counter* each time Authorware passes through the decision structure.

Custom variable names can be followed by an at symbol (@) and the title of an icon. They’re called custom icon variables, and you can use them to find out the custom variable’s value at a specific icon within a file.

Types of variables

Authorware automatically interprets a variable’s type based on how you used it.

Numerical variables store numbers. The numbers can be whole numbers (6), real numbers (0.006), or negative numbers (–33). The numbers stored in a variable can range from -1.7×10^{308} to $+1.7 \times 10^{308}$.

When you add two variables, Authorware treats them as numbers because arithmetic operators can be used only with numbers. Numerical variables might contain a user’s score, a topic number, or the result of a mathematical calculation.

Character variables store character strings. A string is a sequence of one or more characters consisting of letters ("Mary Wagner"), numbers ("49931"), special characters ("*/"), or a combination of any of them ("Total is \$5.00"). A character variable can contain up to 30,000 characters.

Character variables might store a user's name, a list of spelling words, or a URL to a website. When you concatenate two variables, Authorware treats them as character strings because the concatenation operator joins only strings.

Logical variables store a value of `TRUE` or `FALSE`. They're like a light switch, alternating between two states: on and off. The number zero equals `FALSE`, and any nonzero number (usually 1) equals `TRUE`.

Logical variables are particularly useful for activating or deactivating options. For example, inserting a logical variable in a button's Active If field is a good way to activate and deactivate the button. Logical variables are also useful for synchronizing events. For instance, you could use the system variable *Animating* to play a digital movie and move it across the screen at the same time.

When you use a variable in a place where Authorware expects a logical variable (in the Condition field of the Conditional Response Properties dialog box, for instance), Authorware interprets it as a logical variable. If the variable contains a numerical value, Authorware treats the variable as `TRUE` when the value is anything other than zero. If the variable contains a character string, Authorware treats the variable as `TRUE` only if the string is `TRUE`, `T`, `Yes`, or `On` (capitalization isn't important). All other characters are `FALSE`.

List Variables store an array of constant or variable values. Authorware supports two types of lists:

- Linear lists, in which each element is a single value, such as `[1, 2, 3, "a", "b", "c"]`
- Property lists, in which each element consists of a property and a value separated by a colon, such as `[#firstname:"Harry", #lastname:"Smith", #phone:5551212]`

Both types of lists can be unsorted or sorted in alphabetic order.

List variables are particularly useful for keeping track of and updating an array of data.

Symbol Variables A symbol is a type of data like a string or other value that begins with the pound sign (#). Symbols are useful because Authorware can handle symbols more quickly than strings.

For example, the symbol `#Josephine` in the statement:

```
MyVariable := #Josephine
```

runs more quickly than the string "Josephine" in the statement:

```
MyVariable := "Josephine"
```

Variables assigned symbols behave differently than other variables.

For example, after the statement `MyVariable := #Josephine`, embedding *MyVariable* in a display icon will show the result `#Josephine`, because `#Josephine` is the actual value of the variable.

However, the statement `MyVariable + 1` returns a number because in this case, Authorware is performing an arithmetic operation and treats *UserName* as a numerical value. For information about how Authorware handles hierarchies in calculations, see *How Authorware evaluates expressions in Authorware Help*.

Rect Variables A rect is a type of data returned by the Authorware `Rect` system function. Rects are useful for defining rectangular pixel regions. For more information, see `Rect` in Authorware Help.

Point Variables A point is a type of data returned by the Authorware `Point` system function. Points are useful for setting pixel locations. For more information, see `Point` in Authorware Help.

Variables—step-by-step procedures

What you can do	In Authorware Help, see
Use a variable	Using a variable
Display the content of a variable on the screen	Displaying variable information onscreen
Create a custom variable	Creating a variable
Set the initial value of a variable	Creating a variable
Make changes to a custom variable	Changing a custom variable
Delete a custom variable	Deleting a custom variable
Rename a custom variable	Changing a custom variable
Refer to a variable's value at a specific icon	Variable values at a specific icon
Use a variable in a library	Variables, functions, and expressions in libraries

Functions

A function performs a specific task. The task can be as simple as capitalizing the first letter of every word in a line of text or as complex as creating an array.

There are three types of functions: system functions, which are built into Authorware; scripting functions, which you create in Authorware; and custom functions, which include scripting Xtra extensions, U32s, and DLLs. Macromedia supplies a small set of custom functions with Authorware; many more are available from other developers.

The only real difference between system and custom functions is where they come from. They all have the same purpose: to increase the flexibility of Authorware and to extend the range of things you can accomplish.

System functions

You can use Authorware system functions to do such things as manipulate text and files, control and respond to what's going on in navigation structures, play and synchronize media, draw objects and manipulate graphics, jump between files, launch other applications, and perform mathematical functions.

Some functions that complement each other, such as `ArraySet` and `ArrayGet`, come in sets. Some functions work in tandem with variables, such as the `ReadExtFile` function and the system variable `FileLocation`. You can use functions such as `IconNext` and `Keywords` to get information; others such as `CreateFolder`, `DeleteLine`, and `Quit` to carry out tasks; and others such as `Random` and `Round` to perform operations.

System functions in Authorware follow certain naming conventions: They always begin with an uppercase letter and consist of one or more words with no spaces (for example, `Beep` and `VideoSpeed`).

Each system function is described in detail in “System Functions” in Authorware Help.

Getting the syntax right

A function will work right only if you use the correct syntax. The syntax of a function includes the function’s arguments. Almost every system function requires arguments, which supply the function with the information it needs to carry out its task. For example, when you use the `SQRT` function, you need to supply the function with the number you want the square root of.

The most detailed description of a function’s syntax appears in “System Functions” in Authorware Help. Authorware also displays the syntax of each function in the first line of the description in the Functions dialog box.

There are two types of arguments: required and optional. Optional arguments are enclosed in brackets ([]). Not every function has optional arguments.

The best way to get the syntax right is to select a function in the Functions dialog box and then paste where you want to use it. Then replace the argument names in the template syntax with the values you actually want to use. For example, if you want to make sure that the name you’re storing in the variable `UserName` is capitalized, first type:

```
UserName :=
```

Then paste in the `Capitalize` function, like this:

```
UserName := Capitalize(string [,1])
```

Replace *string* with the name of the variable that contains the text the user has entered:

```
UserName := Capitalize(EntryText [,1])
```

And finally, delete the optional argument:

```
UserName := Capitalize(EntryText)
```

If you are entering a function or variable in a Calculation window, you can make sure that the capitalization is correct for case-sensitive functions by setting the properties in the Calculation window to automatically correct the system functions and variables case. This is normally not necessary, however. When you close the calculation icon, the next time you open it, you will see the correct case. For details, see [“Changing calculation defaults in the Calculation window” on page 161](#).

Custom functions in DLLs

Windows dynamic link libraries (DLLs) provide a way of linking to external functions that you call from within calculation icons. Authorware must have the following information about a function in a DLL before it can load it:

- Where the function resides
- The function’s name
- How many and which type of arguments the function takes
- The type of arguments the function returns

User code documents (U32s)

Windows DLLs weren't created for non-programmers. Standard DLL format and calling conventions often require you to refer to programmer documentation and cope with API calls to Windows. To get around that, Authorware supports a transparent extension to the conventional DLL, called a U32. C programmers who create DLLs for use with Authorware can use a U32 to preload the information Authorware needs to recognize and link to functions within the DLL, including function names, arguments, return types, syntax templates, and online descriptions. The additional resources can be stored either with the DLL or separately, and they remain transparent to other programs that have access to the DLL.

When you load a U32, Authorware lists the functions it contains in the Functions dialog box, in the Category whose name is that of your Authorware file. You can select the functions and paste them into a Calculation window as you would a system function. This makes it possible to load and call DLLs without having to use low-level programming conventions.

Conventional DLLs

Authorware also supports conventional DLLs by presenting a dialog box in which you can enter the specific calling parameters. Such conventional support makes it possible to link to DLLs that have not been written to be specifically loadable by Authorware.

DLLs and the search path

Because files for DLLs are separate from the files that call them, they must be in the folders specified in the search path. If Authorware can't locate the custom function in any of these folders, it ignores the function and continues running the piece. Always make sure that the folder containing DLLs is included in the search path.

Loading a DLL custom function

Before Authorware can use a custom function, the function needs to be in memory. Loading a custom function is like opening an Authorware file: Authorware reads the function from the disk where it's stored into the computer's memory. Once the function is there, Authorware can exchange information with the function and tell it when to start. Loading a function connects Authorware with the function and allows the two to communicate. You can start the process of loading a custom function from the Functions dialog box, a Calculation window, or a text object.

When a function is loaded, it takes up space in memory. If you're not using a function in the piece you're working on, you can remove it from memory and free up the space it occupies.

Using functions—step-by-step procedures

What you can do	In Authorware Help, see
Use a system function	System functions
Use scripting Xtra extension functions	Using scripting Xtra functions
Use a custom function	Loading a DLL custom function
Unload a custom function	Unloading a DLL custom function
Use DLLs	Custom functions in DLLs
Use a DLL that doesn't follow Authorware conventions	Loading DLLs that don't follow Authorware conventions
Use functions in libraries	Variables, functions, and expressions in libraries

Expressions and operators

To achieve the results you want with a variable or function, you often need to make the variable or function part of an expression. An expression is a statement that produces a result by performing a calculation or carrying out an operation such as placing a value in a variable. You can use expressions in Calculation windows and dialog boxes, and you can embed them in text objects. In addition to variables and functions, you can use operators, numbers, character strings, constants, and comments in an expression.

For example, this expression increases the value of the variable *Counter* by 1:

```
Counter := Counter + 1
```

This expression uses the *GetLine* function to get the phone number that's stored on the fifth line of the variable *ContactInfo* and put it in the variable *PhoneNumber*:

```
PhoneNumber := GetLine(ContactInfo,5)
```

To tell Authorware what you want it to do with the variables and functions in an expression, you use operators.

Types of operators

An operator is a symbol that stands for the action you want Authorware to perform on one or more values. For example, a plus sign (+) tells Authorware to add two values together. The concatenation operator (^) tells Authorware to fuse two strings of text into one: "car"^"pet" becomes *carpet*. Authorware has five types of operators: assignment, relational, logical, arithmetic, and concatenation. The operators you can use in Authorware are summarized in the following table.

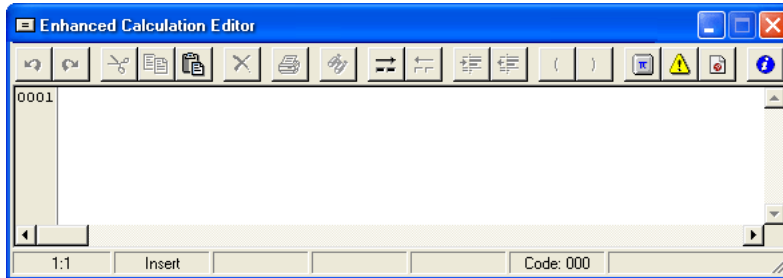
All the operators are described in detail in “System Functions” in Authorware Help. The Language category in the Functions dialog box also contains a complete list of the operators you can use.

Category	Operator	What it means
Assignment operator	:=	Assign the value on the right of the operator to the variable on the left
Relational operators	=	Equal to
	<>	Not equal to
	<	Less than
	>	Greater than
	<=	Less than or equal to
	>=	Greater than or equal to
Logical operators	~	Not
	&	And
		Or
Arithmetic operators	+	Add
	-	Subtract
	*	Multiply
	**	Exponent
	/	Divide
Concatenation operator	^	Join (combines two strings into one)

Code Snippets

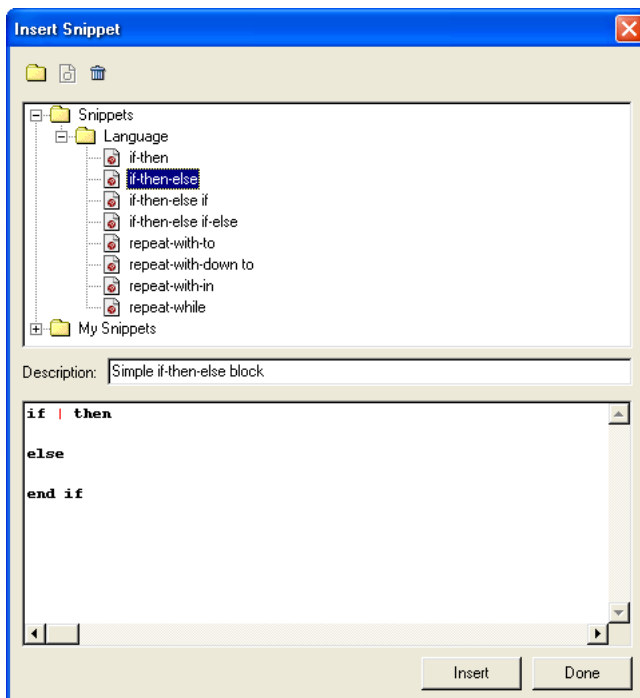
Code snippets are blocks of Authorware code, such as an if...then statement, a database query, or any block of Authorware script that you want to paste into a calculation.

There are three ways to access the code snippets. The first is to click the Insert Snippet button in the Enhanced Calculation Editor toolbar.



Note: You manage the toolbar buttons in the Preference dialog box of the Enhanced Calculation Editor, under the Toolbar tab.

The second way to access code snippets is to right-click in the Calculation window and select Insert > Snippet to open the Insert Snippet dialog box.



The third way to insert a code snippet is by using the keyboard shortcut Control+Y in the Calculation window.

To insert a code snippet:

- Select the name of the snippet and click Insert.

You can also double-click the code snippet name to insert it. Authorware inserts the code snippet into the calculation at the position of the caret.

Note: If the caret is at an indented position, Authorware inserts the snippet with the same indentation level.

When creating custom snippets, you can determine where the caret is placed when Authorware inserts the snippet. The pipeline character (|) determines where the caret appears. Add one to any location in your snippet. If you do not include a pipeline character, the caret is placed at the end of the snippet. If the snippet contains more than one pipeline character, the caret is placed at the last instance.

To add your own snippets to the list:

- 1 Open the Code Snippets dialog box, either by clicking the Snippets icon on the Enhanced Calculation Editor toolbar or by right-clicking in the Calculation window and selecting Insert > Snippet.
- 2 Select an existing folder or create a new one by clicking on the folder icon at the top of the Snippets dialog box.
- 3 Click the New Snippet button at the top of the Snippets dialog box.
- 4 Type a name and description for the new snippet.
- 5 Type or paste the code into the code window at the bottom of the Snippets dialog box.

All the code snippets are stored in a snippets.xml file in the Authorware application directory. You do not need to include this file when you publish your piece.

To delete a code snippet:

- Select a code snippet or a folder, and click on the trash can icon in the toolbar.

Scripting

Scripting combines variables, functions, and expressions—along with if...then statements and repeat loops—to tell Authorware exactly what you want it to do. For more information, see [“Scripting—step-by-step procedures” on page 177](#) for specific procedures.

Here’s an example of a simple script:

```
If StudentScore > 90 then StudentLevel := "expert"
```

The script says, “If the student’s score is greater than 90, then set the student’s level to expert.” It’s constructed of two expressions, `StudentScore > 90` and `StudentLevel := "expert"`, used as the components of an if...then statement.

You can create scripts that are quite complex as well. The following script uses if...then statements nested inside repeat loops to carry out a task over and over in many different places:

```
BranchList := IconID@RootIcon^Return
repeat while LineCount(BranchList) > 0
  SetIcon := IconFirstChild(GetLine(BranchList, 1))
  if IconType(SetIcon) = 4 | =5 | =6 | 9 =|10 | =12 then
    BranchList := BranchList^SetIcon^Return
  end if
  Movable@SetIcon := FALSE
  repeat with x := 1 to IconNumChildren(GetLine(BranchList, 1))
    SetIcon := IconNext(SetIcon)
    if IconType(SetIcon) = 4 | =5 | =6 | 9 =|10 | =12 then
      BranchList := BranchList^SetIcon^Return
    end if
    Movable@SetIcon := FALSE
  end repeat
  BranchList := DeleteLine(BranchList, 1)
end repeat
```

Two of the most common ways to control what happens in Authorware—as the foregoing examples show—are if...then statements and repeat loops. Both are simple enough to master easily, but they're also flexible enough to achieve sophisticated results.

If...then statements

Use an if...then statement to set up a condition and provide Authorware with a specific set of instructions to follow if the condition is TRUE.

In everyday life, you use the equivalent of if...then scripts all the time. You might say, for example, “If it snows, then I’ll go skiing; otherwise, I’ll clean the garage.” Depending on how much you want to go skiing, you could set up additional conditions, such as “If it snows and I finish my taxes, then I’ll go skiing.”

You can use similar statements to determine what Authorware does, checking for one or more conditions.

Checking for more than one condition

You can also check for two or more different conditions and have Authorware do something based on the one that’s TRUE. The syntax looks like this:

```
If condition 1 then
  statement
else if condition 2 then
  statement
else
  statement
end if
```

This syntax tells Authorware, “If this first condition is TRUE, do this action. If this second condition is TRUE, do this action instead. Otherwise, do this third action.”

For example, the following example counts the number of students with scores falling in the high, middle, and low ranges:

```
If PercentCorrect < 66 then
  LowScore := LowScore + 1
else if PercentCorrect < 90 then
  MidScore := MidScore + 1
else
  HighScore := HighScore + 1
end if
```

Repeat loops

Use a repeat loop to repeat an action either for as long as a condition is `TRUE` or for a given number of times.

Using a counter

The simplest type of repeat loop uses a counter to repeat an action a set number of times. The repeat loop can count either up or down, and you can set the starting and ending values of the count with a constant, a variable, or an expression.

In the following example, Authorware adds 1 to *Count*, beeps, and then returns to the beginning of the loop. When the value of *Count* reaches 5, Authorware beeps, then exits the loop.

```
repeat with i := 1 to 5
  Beep(1222, 2000)
end repeat
```

Using a list

You can also set up a repeat loop so that each of the statements in the loop is repeated with an element in a list. For example, the following repeat loop adds the values in *List* together. The value of *X* at the end is 60.

```
List := [10,20,30]
Total := 0
repeat with X in List
  Total := Total + X
end repeat
```

Instead of using a counter, the repeat loop uses the number of elements in the list to determine how many times to repeat the loop.

Using a condition

You can also use a repeat loop to repeat an action as long as a condition is `TRUE`. Each time Authorware repeats the loop, it checks the condition. When the condition changes, Authorware exits the loop.

For example, the following statements find an empty line in the string *TestString*:

```
LineNumber := 1
repeat while GetLine(TestString, LineNumber) <> ""
  LineNumber := LineNumber + 1
end repeat
```


Authorware first sets the value of the variable *LineNumber* to 1. It then gets the first line in *TestString* and checks to make sure it isn't empty (<>" means "is not equal to empty"). If the line isn't empty, it adds 1 to *LineNumber*, then returns to the top of the loop and starts over.

Scripting—step-by-step procedures

What you can do	In Authorware Help, see
Use a variable	Using a variable
Create a custom variable	Creating a variable
Refer to a variable's value at a specific icon	Variable values at a specific icon
Create an expression	Expressions and operators
Use numbers in expressions	Using numbers in an expression
Use text in an expression	Using character strings in an expression
Use constants in an expression	Using constants in an expression
Use operators in an expression	Using operators in an expression
Use system functions	System functions
Use custom functions	Custom functions in DLLs
Use the functions in a scripting Xtra extension	Xtra extensions
Use DLLs	, Using a DLL custom function, Custom functions in DLLs

Xtra extensions

The multimedia features of Authorware take creativity in many directions. As you think of new uses for Authorware, you may want to add features to Authorware that aren't part of the standard Authorware application. To give you flexibility, Authorware supports the use of Xtra extensions—extensions that add new capabilities.

Macromedia produces a few Xtra extensions, such as the Macromedia Director transitions Xtra and the QuickDraw 3D Xtra, but most are produced by independent developers. The Authorware CD contains a few Xtra extensions from independent developers, such as the Zeus Productions transitions Xtra extensions. For information on other Xtra extensions, visit the Authorware Support Center at www.macromedia.com/support/authorware or select Help > Support. by selecting Help > Support.

Types of Xtra extensions

There are five types of Xtra extensions in Authorware:

Transition Xtra extensions create special effects when Authorware displays or erases graphics. For example, you can dissolve from one graphic to another or have one page slide over another. Transition Xtra extensions supplement Authorware built-in transitions and are listed along with the built-in transitions in the Transitions dialog box. The Director transitions, as well as transitions from g/matter and Zeus Productions, are installed with Authorware. For more information, see [“Using transitions for special effects” on page 151](#).

Sprite Xtra extensions add functional elements to the Presentation window. For example, use the QuickDraw 3D Xtra to display a 3D object that users can rotate, resize, and interact with in other ways. With the Macromedia Control Xtra for ActiveX, you can add a wide variety of ActiveX controls—such as Macromedia Flash animations, interactive charts, HTML browser windows, and pop-up menus. For more information, see [“Sprite Xtra extensions” on page 178](#).

Scripting Xtra extensions provide custom functions you can use just like Authorware built-in system functions. For example, Yamaha’s MidXtra, available from the Authorware Support Center, contains functions you can use to control input and output from MIDI devices. For more information, see [“Scripting Xtra extensions” on page 181](#).

MIX, service, and viewer Xtra extensions provide core components of Authorware functionality; you’ll find many of them in the Xtras folder. They’re installed at the same time you install Authorware so that you can use them during development.

Tool Xtra extensions provide you with additional functionality. They appear on the Authorware Xtra menu. For example, the Convert WAV to SWA Xtra is a tool Xtra extension that converts WAV files to SWA files.

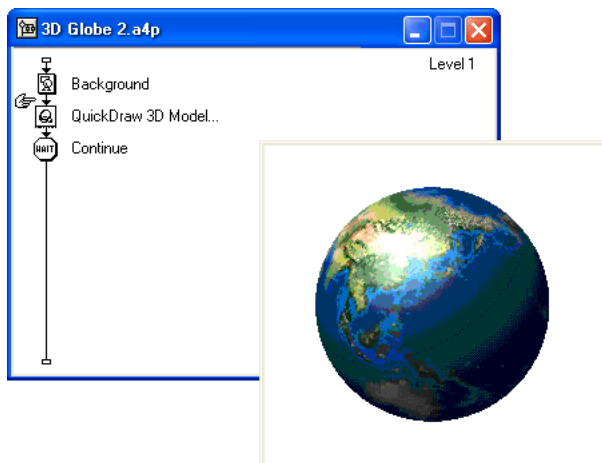
Sprite Xtra extensions

A sprite is a functional element such as a 3D object, a Macromedia Flash animation, an HTML browser window, or a pop-up menu. Sprite Xtra extensions make it possible to incorporate such sprites into an Authorware piece.

When you place a sprite Xtra file in the Xtras folder and then start Authorware, the Xtra file adds a command to the Insert menu. For example, the QuickDraw 3D Xtra adds the command ThreeD to the Insert menu.

Remember that an Xtra file can contain more than one Xtra extension. So the command the Xtra adds to the Insert menu has a submenu that lists all the Xtra extensions in the Xtra file.

To incorporate a sprite into an Authorware piece, position the Paste Hand on the flowline where you want the sprite to appear, then select the Xtra extension you want from the submenu it’s on. Authorware inserts a sprite icon on the flowline. (Some Xtra extensions first display a dialog box that lets you select the sprite you want.) When you run the piece, the sprite appears in the Presentation window.



Sprite icon properties

Like all icons, sprite icons have properties that you can set in the icon's Properties dialog box.

The general properties of a sprite icon—which all sprite icons share—are similar to those of a display icon: where it's positioned in the Presentation window, whether users can move it, the transition you've applied to it, and so on. For more information about the general properties of a sprite icon, see [“Sprite properties” on page 179](#).

Sprite properties, events, and methods

In addition to the general properties that all sprite icons share, each sprite icon can also have a set of unique features. Those features fall into three categories: properties, events, and methods. Not all sprite icons have features in all three categories. For example, a QuickDraw 3D sprite icon has properties but neither events nor methods.

Click the Options button in the Properties dialog box to see the properties, events, and methods available for a sprite icon. The appearance of the options dialog boxes varies considerably. ActiveX controls have their properties, events, and methods organized on separate tabs.

The QuickDraw 3D sprite icon—because it has only properties—has separate tabs devoted to different types of properties.

Details about properties, events, and methods

For specific information about the properties, events, and methods available for a sprite Xtra extension, check the documentation that the developer has provided. The documentation may be available through the help system (click the Help button in the sprite icon's options dialog box to display it), or it may be a Read Me file.

Sprite properties

Properties are characteristics or values. For example, the QuickDraw 3D sprite icon has dozens of properties that affect the rotation and position of the model, how it's rendered to the screen, the direction and quality of the lighting, and the position and orientation of the camera.

Properties can affect things other than a sprite's appearance. For example, the QuickDraw 3D sprite has properties that control the user's ability to manipulate the model. A button sprite might have a property that determines whether it works like a push button or a radio button.

In some cases, both a sprite icon and the sprite it displays in the Presentation window have properties. For example, the properties of the QuickDraw 3D sprite icon act as defaults for the properties of the sprite itself. You can, however, change the properties of the sprite directly—using the `SetSpriteProperty` function—without changing the properties of the sprite icon.

Authorware has four functions you can use to manipulate sprite icons and sprite properties.

Function	Description
<code>GetIconProperty</code>	Gets the value of the asset property of the sprite icon.
<code>GetSpriteProperty</code>	Returns the value of the property for a sprite.
<code>SetIconProperty</code>	Sets the value of the property for the sprite icon's asset.
<code>SetSpriteProperty</code>	Sets the value of the property for the sprite currently displayed by a sprite icon.

When you're working with ActiveX controls, use the `GetSpriteProperty` and `SetSpriteProperty` functions.

Sprite events

Events are exactly what they sound like: things that happen. Some events are produced by users. For example, the Microsoft Calendar control tracks where a user clicks on the calendar. Other events are produced by a sprite on its own. The Microsoft Web Browser control, for example, sends an event when a download begins and another when the download is complete. Sprites that produce events are called senders.

Use the event response type to select a sender, to track the event you want to respond to, and to set up Authorware to respond when it happens. There's also a set of variables and functions you can use to manage events.

Variable	Description
<code>EventLastMatched</code>	Contains a property list for the Xtra extension event matched at the last event response.
<code>EventQueue</code>	Contains a list of all pending unprocessed external events sent by Xtra extensions in order of arrival.
<code>EventsSuspended</code>	If <code>TRUE</code> , prevents all event responses from interrupting the current flow and saves all events in the variable <code>EventQueue</code> .
<code>GlobalTempo</code>	Contains the rate at which sprite Xtra extensions receive step events.
<code>FlushEventQueue</code>	Discards all pending events from the event queue.
<code>SendEventReply</code>	Sends a reply to an event sent by an Xtra extension.

Sprite methods

Methods are similar to custom functions. They let you perform some action to or through a sprite or a sprite icon. For example, use the `GoBack` and `GoForward` methods in the Microsoft Web Browser control to navigate through the pages of a website.

The primary difference between a method and a custom function is that you can't use a method directly in a Calculation window. Instead, use the `CallIcon` function to call a method associated with a sprite icon or the `CallSprite` function to call a method associated with a sprite. (For information about which function is appropriate for the method you want to use, check the documentation that came with the Xtra extension.)

For example, to use the `GoBack` method, set up a statement like this in a Calculation window:

```
CallSprite(IconID@"Web Browser",#GoBack).
```

When you're working with ActiveX controls, always use `CallSprite`.

For more information about the `CallIcon` and `CallSprite` functions, see "System Functions" in Authorware Help.

Scripting Xtra extensions

Scripting Xtra extensions are packages of custom functions. They're a way of incorporating functionality into an Authorware piece that Authorware can't provide on its own. For example, the Yamaha MidXtra lets you control MIDI devices, which you can't do with any of the functions that are built into Authorware.

When you place a scripting Xtra file in the Xtras folder and start Authorware, Authorware adds a category for the Xtra extension to the Functions dialog box.

Selecting the category displays a list of the functions contained in the Xtra extension. The syntax and a description of the selected function appears.

You can then select the function you want to use and paste it into a calculation window.

Global functions, parent objects, and child objects

Global functions are indistinguishable from Authorware system functions. You set them up and use them the same way. For example, the FileIO Xtra's GetOSDirectory function—which gets the path to the Windows directory—is as straightforward as using the Authorware GetLine function.

Parent objects are analogous to the icons on the Icon palette. Each has some well-defined task like displaying images or playing digital movies. When you drag an icon to the flowline, you're creating a child of the parent icon. The child icon is an instance of the parent, with its own unique attributes and behaviors: For example, it plays a specific digital movie at a specific frame and at a specific rate. Likewise, a child object is a self-contained, independent instance of a parent object.

Using parent and child objects

The tasks that a parent object can perform are defined by its methods. (A method, you'll remember, is another name for a custom function.) The MidXtra can, for example, start, stop, and loop sounds, change harmonics, add reverb, and so on. But just as a specific instance of the sound icon on the flowline plays a sound file, a child object—not its parent—provides the specific instance of a method that you use to perform a specific task.

Using parent and child object methods is different from using system functions. You first need to use the Authorware NewObject function to create and name a child object before you can use a child object method to perform a specific task. (A parent object, on the other hand, is created automatically by the scripting Xtra extension; you can't create more of them.)

In the following example, the NewObject function creates a new instance of the MidXtra (the parent object containing the MIDI methods) and names it Intro Music:

```
Intro Music:= NewObject("MidXtra")
```

Once you've created the child object, use the Authorware CallObject function to call the method you want to use. In the following example, CallObject calls the MIDIload method to load the sound file named Intro.aif:

```
CallObject(Intro Music,"MIDIload","Intro.aif")
```

Just as you need to create a child object before you can use it, you need to use the DeleteObject function to get rid of it when you're finished with it.

In most cases you need to create a child object to use a method, but a few methods are exclusive to the parent object. For example, the FileIO Xtra's `version` method returns the version number of the Xtra extension.

To call a parent object method, use the Authorware `CallParentObject` function. For example, to call the FileIO Xtra's `version` method, use this statement:

```
Xtra Version:= CallParentObject("fileio",#version)
```

`CallParentObject` calls the `version` method, which gets the version number of the FileIO Xtra. `CallParentObject` then puts the version number in the custom variable *Xtra Version*.

Note: Parent and child objects in scripting Xtra extensions are the Authorware implementation of object oriented programming. If you're interested in finding out more about how parent and child objects work, take a look at one of the many books available on object oriented programming.

Functions for using parent and child object methods

The following table lists the functions available for working with parent and child object methods, along with a brief description of each one. For more information about them, see "System Functions" in Authorware Help.

Function	Description
<code>CallObject</code>	Calls a child object method.
<code>CallParentObject</code>	Calls a parent object method.
<code>DeleteObject</code>	Deletes a child object created by <code>NewObject</code> .
<code>NewObject</code>	Creates a child object.

Identifying global functions and parent and child methods

Global functions and parent and child methods all look the same when they're listed in the Functions dialog box. To figure out whether an item is a global function, a parent method, or a child method, select it and look at the syntax that appears in the Description field:

- The syntax of a child method begins with `CallObject`.
- The syntax of a parent method begins with `CallParentObject`.
- The syntax of a global function looks just like the syntax of a system function.

ActiveX controls

ActiveX is Microsoft's name for its object linking and embedding (OLE) technology. The idea behind the technology is to allow one application (such as a spreadsheet) to plug into another application (such as a word processor) so that the information you produce in one can be an integral part of the information you produce in the other. The result could be, for example, a spreadsheet embedded in a business plan.

While ActiveX is essentially the same technology as OLE, its intent is slightly different. ActiveX is a streamlined and simplified approach aimed at allowing developers to produce small applications, known as controls, with a highly focused purpose. For example, the Flash control for ActiveX control is designed just to play Flash animations, not to create them.

The Macromedia Control Xtra for ActiveX is a valuable addition to Authorware. Developers have created thousands of ActiveX controls, and the uses for this Xtra extension are as limitless as the variety of ActiveX controls available. Here are just a few examples:

- Use the Microsoft Web Browser control to let users explore the Internet.
- Create complete Windows applications with MicroHelp's extensive library of Windows controls.
- Use the Microsoft Calendar control as an interface to a project log.
- Let users explore virtual worlds with the InterVista VRML control.

ActiveX controls—step-by-step procedures

What you can do	In Authorware Help, see
Use an ActiveX control	Using an ActiveX control
Install an ActiveX control	Installing ActiveX controls
Install the Macromedia Control Xtra for ActiveX	Installing the Macromedia Control Xtra for ActiveX
Install the system files users need to run ActiveX	Installing ActiveX system files
Install and register an ActiveX control	Installing and registering the ActiveX controls

The XMLParser Xtra

The XML Parser Xtra allows Authorware pieces to read, parse, and make use of the contents of XML documents. To use the XMLParser Xtra, you must know something about the structure and content of the documents you are parsing.

Note: Because some of the functionality has been changed, the XMLParser Xtra for Authorware is a different product from the XMLParser Xtra for Director. They cannot be interchanged.

About XML

Although XML and HTML documents appear similar in some ways, only their syntax makes them look similar. HTML is a language for page layout, and XML is a language for the representation of data that is used to store and transmit text data. XML is a specification for how data gets marked up; it does not specify which names are used to tag elements. So for each application, the programmer can develop whatever structure and elements are appropriate. Unlike HTML, XML says nothing about how it should be displayed (or *rendered*). Although MS IE5+ has a plug-in to display an XML file, it merely shows the hierarchy itself in a TreeView format.

One of the main differences between the languages is that XML must have both starting and ending tags for all elements:

```
<tag attr1="blah">  
  content  
</tag>
```

For elements that do not contain content, there is an abbreviated syntax called an empty element tag:

```
<tag attr1="blah"/>
```

XML has become popular because of its simplicity and flexibility. It is also self-documenting, because the element names (tags) and attribute names describe what is being stored. The redundancy of naming is considered a small price to pay for the ease of maintaining XML. XML files can easily represent hierarchical structures. Also, a given level can contain multiple elements of the same type.

The earlier technology that XML can be best compared with is INI files. In fact, most modern applications now use XML files to store preference settings rather than INI files. See Using the XML Parser Xtra in Authorware Help.

CHAPTER 9

Tracking Performance and Working with a Learning Management System

When you create a training course, you naturally want to track how well users perform and keep a permanent record of their performance. Tracking users' performance has several benefits:

- You can provide users with feedback as they work through the course.
- When they finish the course, you can assess how well they've done.
- You can use information about their performance to design a more effective training piece.

Macromedia Authorware makes tracking performance easy because everything you need is built in. When you set up a target response in an interaction, you need only a single setting to indicate whether the response is right or wrong. Authorware tracks users' performance automatically—and in some detail—with a wide range of built-in variables. It keeps track of how many right and wrong target responses a user has made, how many tries the user took to get the right answer, how much time it took the user to respond to an interaction, how much time the user spent in an interaction, how much time remained if there was a time limit on an interaction, any objects the user has clicked or moved, and any words or lines the user has clicked.

Once you've collected the information you want, you can display the information on the screen or use it to determine what feedback the user needs. You can use Knowledge Track to exchange standard student performance information with a learning management system. Or you can use the Authorware built-in computer-managed instruction (CMI) functions to custom-tailor the student information you exchange with your learning management system. You can also use the ODBC capabilities of Authorware to save the information in a database such as Access.

Using Assessment Knowledge Objects

You may also want to use the Authorware Assessment Knowledge Objects to use a wizard procedure to set up your assessments. For a list of the Assessment Knowledge Objects and how they are used, see [“Assessment Knowledge Objects” on page 39](#).

QTI Assessment Knowledge Objects

Several Assessment Question Knowledge Objects have been updated to allow for import and export of question content stored in XML files that conform to the IMS QTI (Question Test Interoperability) version 1.1 standard. The following Assessment Knowledge Objects have been updated:

- Hot Spot Question
- Hot Object Question

- True/False Question
- Single Choice Question
- Multiple Choice Question

For details, see [How to set up Assessment Knowledge Objects for QTI in Authorware Help](#).

The IMS QTI standard is still evolving. For more information about the standard, refer to www.imsglobal.org.

Also check the Macromedia Authorware Support site at www.macromedia.com/support/authorware for updates and any late-breaking information about the IMS QTI support in these Knowledge Objects.

Authorware Learning Object Content Packager command

The Authorware Learning Object Content Packager lets you package SCORM-compliant learning objects for web delivery. For more information about the Learning Object Content Packager, select **Commands > Authorware Learning Object Content Packager** and press F1.

Authorware Learning Object Metadata Editor

The Authorware Learning Object Metadata Editor provides the functionality needed to meet Advanced Distributed Learning (ADL) Shareable Content Object Reference Model (SCORM) requirements to create an XML file with content metadata.

For more information about the Metadata Editor, select **Commands > Authorware Learning Object Metadata Editor** and press F1 to get specific help about how the editor is used.

Guidelines for tracking and recording performance

As you plan what you want to track in a project and how you want to record performance, here are some things to think about:

- What activities do you need to track to assess a user's performance?
[“Tracking performance” on page 188](#) describes the activities you usually need to monitor to assess performance. This section also lists all the variables available in Authorware for tracking what a user is doing.
- How are you going to use the information you gather? Are you going to use it to provide feedback? Display it on the screen? Save it permanently in a text file or database?
 For details, see [Providing Feedback and Tracking and recording performance—step-by-step procedures in Authorware Help](#).
- Where are you going to store performance information? On the user's computer? On a local server? On a web server across the Internet?
 For more information, see [“Recording performance” on page 191](#).
- Do you want to store the information you've gathered in a database?
 For details, see [“Connecting with databases” on page 200](#).

Marking a response right or wrong

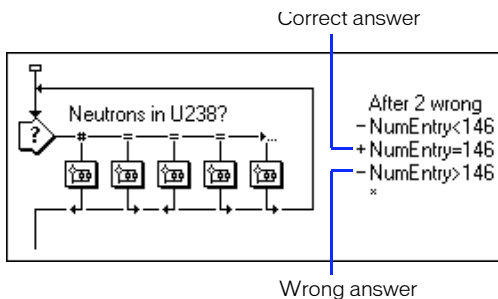
A response—clicking a Help button, for example, or entering your name—is neither right nor wrong. If you want Authorware to track a user's performance, you need to mark the target responses in an interaction as correct or incorrect.



Use the Status setting to mark a response as right or wrong.

An interaction that contains a target response you've marked is called a judged interaction.

When you select Correct Response or Wrong Response, a symbol appears next to the result icon's title to indicate that the response has been judged.



A plus sign (+) indicates a correct answer; a minus sign (–) indicates a wrong answer.

You can mark more than one target response in an interaction correct. For example, if you set up an interaction in which students drag certain animals into an environment to create a balanced ecosystem, there might be half a dozen correct answers.

Marking a target response in an interaction as either right or wrong only instructs Authorware to keep track of how many correct target responses and how many wrong target responses a user matches in a single session. Authorware automatically stores that information in a special set of system variables. You still need to create appropriate feedback for the target response and select the exit branching option that you want to use.

Tracking performance

Authorware uses system variables to automatically track a wide range of information about a user's performance. Here are some specific examples of how variables can provide a rich learning environment for the people you're training.

Tracking response time

The system variable `ResponseTime` tracks the length of time it takes a user to respond to an interaction. This variable is useful for limiting the amount of time a user has to respond in an interaction. For example, you could set up a calculation that automatically displays a warning if `ResponseTime` is greater than 15 seconds, and another calculation that exits the interaction automatically when `ResponseTime` reaches 30 seconds.

The value of `ResponseTime` is set to 0 each time a user responds, so it isn't a good choice if you want to limit the total time a user spends in an interaction but also give the user more than one try. In that case use the system variable `TimeInInteraction`.

Tracking time spent in an interaction

The system variable `TimeInInteraction` tracks the total time a user spends in an interaction. The time is measured from the moment Authorware enters the interaction until the moment it exits the interaction. You can use `TimeInInteraction` to limit the amount of time users have in an interaction when you're giving them more than one try or when the correct answer is made up of more than one response.

Tracking time spent in a section

You can use the system variable `SystemSeconds` to calculate how long a user has spent in a section of a course. `SystemSeconds` contains the number of seconds that have elapsed since a user turned on or restarted the computer. If you use a custom variable to record the value of `SystemSeconds` at the beginning of the section, you can subtract that value from the value of `SystemSeconds` at the end of the section to figure out how long the user has spent in the section.

This variable can be useful in determining how quickly a student learns. You might use the result to adjust the difficulty of a course to suit the learning speed of different students.

You can also use `SystemSeconds` to limit the amount of time a user has to complete a section.

Tracking time remaining

The system variable `TimeRemaining` keeps track of how much time is left in an interaction that has a time limit. Use it to give users feedback at appropriate points as time runs out.

Tracking the number of tries

The system variable `Tries` keeps track of the number of times a user responds to a particular interaction. You might give increasingly specific hints as the value gets bigger.

Tracking the choices made

You can use the system variable `ChoicesMatched` to keep track of how many different responses a user has tried in an interaction. This is a good diagnostic tool for determining how well a user knows the material and it's also useful for tailoring feedback to a user who's having particular trouble with a question.

Identifying objects clicked, moved, or matched

The `ObjectClicked`, `ObjectMatched`, and `ObjectMoved` system variables keep track of objects that a user has interacted with.

When a user clicks an object in the Presentation window, Authorware puts the name of the display icon that contains the object in the system variable `ObjectClicked`. Use `ObjectClicked` to give users feedback before they move the selected object. For instance, display a message such as “Now drag the part to its proper position.”

`ObjectMatched` contains the name of the last object a user moved to the correct target area in an interaction. `ObjectMoved` contains the name of the last object a user moved, whether or not it was moved to the correct target area. You can use both in a target area interaction where you've selected the Match Any Object option to accept any object that a user drags to the target area. For example, use `ObjectMatched` to check which object the user moved into the target area, and then provide appropriate feedback. Use `ObjectMoved` to handle cases in which the user moved an object but didn't move it into the target area.

Identifying the word or line a user clicks

The system variables `LineClicked` and `WordClicked` keep track of text a user has clicked.

When a user clicks a line in a list or a block of text, `LineClicked` contains the number that corresponds to the line's position in the list or text block. When a user clicks a word that's part of a text object, Authorware puts the word in `WordClicked`.

You could use both `LineClicked` and `WordClicked` to test users' ability to understand the readout on the screen of a monitoring device. For example, you could ask them to click the line that displays the pressure in a vessel and the word that indicates the status of a relief valve.

Note: The system variable `HotTextClicked` also identifies a word that a user has clicked. But it's designed to work with words you've set up as hypertext links—words that, when clicked, take a user from one part of an Authorware piece to another. For more information, see [“Navigating with hypertext” on page 148](#).

Other variables for tracking

There are several other variables you can use to track performance. They're explained in detail in *System Variables in Authorware Help*.

Variable	Description
<code>AllCorrectMatched</code>	TRUE if a user has matched all the target responses you've marked as correct.
<code>ChoiceNumber</code>	Contains the number of the last response type symbol whose target response the user matched.
<code>ChoicesMatched</code>	Contains the number of different response type symbols whose target response the user matched.

Variable	Description
CorrectChoicesMatched WrongChoicesMatched	Contains the number of different target responses a user has matched in an interaction.
FirstTryCorrect* FirstTryWrong*	Contains the total number of judged interactions that a user answered right or wrong on the first try.
JudgedInteractions*	Contains the total number of judged interactions a user encountered in a file.
JudgedResponses*	Contains the total number of responses a user made to all the judged interactions encountered in a file.
LastLineClicked	Contains the number of the line a user clicked in a text object.
LastObjectClicked	Contains the title of the display icon that contains the last object a user clicked.
LastParagraphClicked	Contains the paragraph number of the last paragraph a user has clicked.
LastWordClicked	Contains the last word a user clicked in a text object.
MatchedEver	TRUE if a user has previously matched the same target response the user has just matched.
MatchedIconTitle	Contains the title of the last response type symbol whose target response a user has matched.
ParagraphClicked	Contains the number of the paragraph a user clicked in a text object.
PercentCorrect PercentWrong	Contains the percentage of all the judged responses that a user answered right or wrong.
PreviousMatch@~ IconTitle	TRUE if the response type symbol named in IconTitle was the last one matched by a user.
ResponseStatus	Contains the status of the first judged response type symbol that a user matched in an interaction.
TimesMatched	Contains the number of times a user matched the current response type symbol during the current session.
TotalCorrect* TotalWrong*	Contains the total number of right or wrong responses a user made to all the judged interactions encountered in a file.

Providing feedback

Feedback is an important part of an effective course. One of the easiest ways to provide feedback—and one of the simplest ways to use variables—is to display the value of a variable on the screen.

Feedback lets users track their own performance and see their strengths and weaknesses. For those sections of the course that build on what users learned in previous sections, a performance summary can help users to prepare. For example, knowing that they did well in one segment prepares and motivates users for working with additional information.

To provide a user with a performance summary, display the value of variables such as these:

- PercentCorrect, PercentWrong
- TotalCorrect, TotalWrong

- FirstTryCorrect, FirstTryWrong
- JudgedInteractions
- JudgedResponses

Say that at the end of a course you want to display a summary that looks like this:

Out of 25 responses to 20 questions, you answered 85% correct.

You answered 15 questions right on the first try.

To create this summary, set up a text object like this:

Out of {JudgedResponses} responses to {JudgedInteractions} questions, you answered {PercentCorrect}% correct.

You answered {FirstTryCorrect} questions right on the first try.

Recording performance

Authorware keeps track of information about a user's performance automatically, and it also collects information that you ask the user to enter, such as the user's name or password. However, Authorware doesn't store this information indefinitely, and it doesn't save the information permanently before it quits unless you tell it to.

It's good practice to store information, as soon as Authorware has collected it, in a custom variable where it won't be overwritten as Authorware continues to gather information.

If you want to keep the information permanently, save it in an external file—a data file saved on a disk.

Storing information in Authorware

Whenever a user enters something in a text-entry field, Authorware automatically stores the information. It stores text in the system variable `EntryText` and numerical values in the system variables `NumEntry`, `NumEntry2`, and `NumEntry3`. But because Authorware stores the information automatically, it also automatically overwrites the information as soon as the user enters new information.

Suppose you ask a user to enter her name and then her password. When she enters her name, Authorware stores it in `EntryText`. Then, as soon as she enters her password, Authorware stores it in `EntryText`, overwriting her name. Fortunately it's very easy to transfer the user's name to a different variable before she enters her password. You can do that with a simple assignment statement, in this case:

```
UserName := EntryText
```

To keep a permanent record of information a user has entered—rather than just storing it temporarily while Authorware is running—save the information in an external file or database. See [“Saving information in an external file” on page 192](#) and [“Connecting with databases” on page 200](#).

Saving information in an external file

You can save information that users have entered by transferring it from the variable where Authorware has stored it to an external file.

Usually you'll want to use a variable to name the file where you're storing user information so you can change the filename for each user. One of the simplest approaches is to ask a user to enter his or her name at the beginning of a piece, store the name in the system variable `UserName`, and then use `UserName` as the name of the file where you store information about a user's performance. For more information, see the `UserName` function.

Saving information on a local hard disk or server

To save information on the user's hard disk or on a local server, use the Authorware `WriteExtFile` and `AppendExtFile` functions. It's best to save files either in the records folder or in the folder containing the Authorware file the user is running. Authorware creates the records folder automatically to give you a convenient place to save information. The records folder is named `A7w_data`, and it's located inside the Windows or Winnt folder, depending on which operating system you are using. You can change the location of the records folder with the `JumpFile` and `JumpFileReturn` functions. You can also set the location of the records folder in the `Aw7.ini` file.

There's a good reason to save files either in the records folder or in the folder containing the Authorware file the user is running: Authorware keeps track of the paths to those folders and stores them in the system variables `RecordsLocation` and `FileLocation`. When you use either variable, all you need to specify is the location inside the records folder or the Authorware file folder where you want to save a file—Authorware fills in the rest of the path automatically. For more information about these variables, see “`FileLocation`” and “`RecordsLocation`” in Authorware Help.

Saving information over the Internet

Transferring information over the Internet—or over a corporate, government, or educational intranet—differs in two significant ways from saving information on a user's hard disk or on a local server:

- You can't use the Authorware `WriteExtFile` and `AppendExtFile` functions. Instead, you use the equivalent functions `FtpStore` and `FtpAppend` that are part of the FTP U32.
- You can transfer information with a web-packaged piece only if a user has chosen trusting mode. Users can select non-trusting mode to protect their system from web-packaged pieces that might read confidential files from their hard disk and transmit them back to the server. For more information, see “Authorware Web Player security features” in Authorware Help.

Managing information with a database

An external file and the Authorware records folder are fine for simple record keeping, but if you want to manage information with any sophistication, you need a database.

For example, if you're creating training for a company that requires employees to complete regular training courses, you probably need to set up a system that constantly updates the records at the end of courses and provides an accurate list of who has completed which courses.

For more information about using databases, see [“Connecting with databases” on page 200](#).

Tracking and recording performance—step-by-step procedures

What you can do	In Authorware Help, see
Mark a response right or wrong	Marking a response right or wrong
Display a user's score on the screen	Displaying a user's score
Store information a user enters	Recording information a user enters
Save information a user enters permanently	Saving user information
Store text and values	Storing text and values
Use <code>WriteExtFile</code> to save information	Setting up <code>WriteExtFile</code> to save user information
Set the path for the file you're saving information in	Setting up <code>WriteExtFile</code> to save user information
Name the file you're saving information in	Setting up <code>WriteExtFile</code> to save user information
Save the content of a variable in a file	Setting up <code>WriteExtFile</code> to save user information
Retrieve information from a file	Retrieving information
Display information from a file on the screen	Retrieving information
Use <code>ReadExtFile</code> to retrieve information from a file	Retrieving information

Using Knowledge Track to set up a course

The best way to keep track of student performance is to use an AICC-compliant computer-managed instruction (CMI) system. This type of management server is also known as a learning management system (LMS).

Authorware makes it easy for you to pass student performance data to a CMI system with:

- Knowledge Track settings for files, interactions, and responses
- A comprehensive set of tracking variables and functions

Use the Knowledge Track settings in the File Properties, Interaction Properties, and Response Properties dialog boxes to track basic student performance data. If you need to do more sophisticated tracking, use the Authorware learning management system variables and functions to customize the way Authorware gathers data and passes it to your learning management system.

Knowledge Track settings

The Knowledge Track settings in the File Properties, Interaction Properties, and Response Properties dialog boxes make it easy to gather basic student performance data such as scores, the amount of time students spend in both individual interactions and entire courses, and the specific answers each student gives (not just whether a student's answers are right or wrong).

The following tables provide an overview of the information you can gather with the Knowledge Track settings.

CMI tab (File Properties dialog box)

Setting	What it does
All Interactions	Turns on tracking for all the interactions in the file.
Score	Tracks and adds up a student's score.
Time	Tracks how long a student takes to complete all the interactions in the file.
Timeout	Tracks a student's inactivity. If too long a period passes during which a student does nothing, Authorware takes special action.
Logout Upon Exit	Logs students out of the learning management system when they quit Authorware.

CMI tab (Interaction Properties dialog box)

Setting	What it does
Interaction	Turns on tracking for the interaction (as long as All Interactions is checked in the File Properties dialog box).
Interaction ID	Specifies the interactions's unique identifier in the learning management system.
Objective ID	Specifies the objective ID the interaction is tied to in the learning management system.
Weight	Indicates the relative importance of the interaction.
Type	Indicates the type of interaction, multiple choice or fill-in.

In addition to the information you enter in the dialog box, Authorware sends the date and time the student exited the interaction and the number of seconds the student spent in the interaction. Authorware also sends whether or not the student answered correctly and the answer the student chose or entered.

Response Properties dialog box

Setting	What it does
Score	Sets the value associated with the response.

CMI variables and functions

The Authorware computer-managed instruction (CMI) variables and functions are listed in the following tables.

Variables

For detailed information about the following variables, see the system variables overview in [Authorware Help](#).

Variable	Description
CMIAAttemptCount	Contains the number of times a student has visited the activity.
CMIAAttempts	Contains a list of property lists to store attempt information.
CMICompleted*	Indicates whether a student has completed an activity.
CMIconfig	Contains configuration information entered in the learning management system.
CMICourseID	Contains the course number entered in the learning management system.
CMIData*	Exchanges miscellaneous activity data with the learning management system.
CMIFailed*	Indicates whether a student has failed an activity.
CMILoggedOut*	Logs the student out from the learning management system immediately after exiting the activity.
CMIMasteryScore	Contains the required mastery score for the activity.
CMIObjCount	Contains the number of objectives related to the activity.
CMIObjectives	Contains a list of property lists to store objective information.
CMIPassed*	Indicates whether the student has passed the activity.
CMIPath	Contains the full path to the private data directory for the student.
CMIReadComplete*	Deletes temporary data files immediately after the data is transferred between the learning management system and Authorware.
CMIScore*	Contains the score for the activity.
CMISTarted*	Indicates whether the student has started the activity.
CMISTatus*	Indicates the status of the activity, "Completed," "Incomplete," or "Not Attempted."
CMITime*	Contains the number of seconds a student has spent on the activity.
CMITimedOut*	Logs the student out of the learning management system if too much inactive time has passed during the activity.
CMITrackAllInteractions*	Setting <code>CMITrackAllInteractions</code> to <code>TRUE</code> allows a learning management system to track the interactions in a file.
CMITrackInteraction*	Applies to a specific interaction icon.
CMIUserID	Contains the student's unique identification string.
CMIUserName	Contains the user name that the student used to log in to the learning management system.
TotalScore	Contains the sum of the scores of all the student's responses.

Functions

For detailed information about the following functions, see the system functions overview in [Authorware Help](#).

Function	Description
CMIAAddComment	Adds an activity-specific comment to the CMI course.
CMIAAddInteraction	Transfers specific information about an interaction to the learning management system.
CMIAAddInteractionEx	Transfers specific custom data about an interaction to the learning management system.
CMIFinish	Sends all finalized tracking data to the server. Sends <code>ExitAU</code> .
CMIFlush	Sends all current tracking data to the server.
CMIGetAICCVersion	Returns the current AICC version. The version can be set to any string. For details, see "CMISetAICCVersion" on page 197 .
CMIGetAttempt	Returns from the learning management system the number of times the student has attempted the activity.
CMIGetAttemptScore	Returns the score for an activity attempt given an attempt number between 1 and the value of the <code>CMIAAttemptCount</code> .
CMIGetAttemptStatus	Returns a status for an attempt given an attempt number between 1 and the value of the <code>CMIAAttemptCount</code> system variable.
CMIGetConfig	Returns activity configuration data specified in the learning management system.
CMIGetCourseID	Returns the course number specified in the learning management system.
CMIGetCustomField	Returns the value stored in the custom field indicated by the table and field names.
CMIGetData	Returns activity data specified in the learning management system.
CMIGetDemographics	Returns the student demographics information indicated by the specified field name.
CMIGetLastError	Use to determine whether an error occurred in the last CMI operation.
CMIGetLocation	Returns from the learning management system the last place the user left off in the activity.
CMIGetMasteryScore	Returns the required mastery score for the activity as set in the learning management system.
CMIGetObjCount	Returns the number of objectives related to the current activity.
CMIGetObjID	Returns the unique identifier for an activity objective specified by an index number between zero and the value of the <code>CMIObjCount</code> system variable.
CMIGetObjScore	Returns the current score for the activity objective specified by an index number between 1 and the value of the <code>CMIObjCount</code> system variable.
CMIGetObjStatus	Returns the status of the activity objective specified by an index number between 1 and the value of the <code>CMIObjCount</code> system variable.
CMIGetPath	Returns from the learning management system the path to the private student directory.

Function	Description
CMIGetScore	Returns from the learning management system the numeric score for the activity.
CMIGetStatus	Returns from the learning management system the status of the current activity as the user last left it.
CMIGetTime	Returns from the learning management system the total time in seconds that the user has spent in the activity.
CMIGetUserID	Returns the student's unique identification string as set in the learning management system.
CMIGetUserName	Returns the user name that the student used to log in to the learning management system.
CMIInitialize	Allows the learning management system to perform any initialization necessary before running the activity.
CMIIsAttemptCompleted	Returns whether the student completed the specified attempt.
CMIIsAttemptFailed	Returns whether the student failed the specified attempt.
CMIIsAttemptPassed	Returns whether the student passed the specified attempt.
CMIIsAttemptStarted	Returns whether an attempt is started given an attempt number between 1 and the value of the CMIAttemptCount system variable.
CMIIsCompleted	Returns TRUE if the learning management system indicates that the current activity is complete, FALSE otherwise.
CMIIsFailed	Determines whether or not the student failed the current activity.
CMIIsObjCompleted	Returns whether an objective is completed given an objective index between 1 and the value of the CMIObjCount system variable.
CMIIsObjFailed	Returns whether an objective is failed given an objective index between 1 and the value of the CMIObjCount system variable.
CMIIsObjPassed	Returns whether an objective is passed given an objective index between 1 and the value of the CMIObjCount system variable.
CMIIsObjStarted	Returns whether an objective is started given an objective index between 1 and the value of the CMIObjCount system variable.
CMIIsPassed	Determines whether or not the student passed the current activity.
CMIIsStarted	Returns TRUE if the learning management system indicates that student has started but not completed the current activity.
CMILogin	Logs the student specified by the sign-on name and password into the CMI server. May not work with all Learning Management Systems.
CMILogout	Logs the student out of the current activity. May not work with all Learning Management Systems.
CMIReadComplete	Should be called after obtaining all the information from the learning management system.
CMISetAICCVersion	Sets the client AICC version number for compatibility with different CMI systems. The version can be set to any string. The default is version 2.0.
CMISetCompleted	Sets the current activity as completed.

Function	Description
CMISetCustomField	Sets a value in the custom field indicated by the table and field names.
CMISetData	Passes miscellaneous activity data to the learning management system.
CMISetFailed	Sets the current activity as failed.
CMISetLocation	Sets where the student last left off in the current activity. Accepts up to 255 characters.
CMISetLoggedOut	Sets the current activity as logged out.
CMISetObj	Transfers specific information about an objective to the learning management system given an objective index between 1 and the value of the CMIObjCount.
CMISetPassed	Sets the current activity as passed.
CMISetScore	Passes the numeric activity score to the learning management system.
CMISetStarted	Sets the current activity as started.
CMISetStatus	Passes the current activity status to the learning management system.
CMISetTime	Sets the total number of seconds the student has spent in the activity.
CMISetTimedOut	Sets the activity as timed out.

Learning Management System (LMS) examples

In the Authorware program file Goodies folder is a another folder called LMS examples. These examples show you how you can use Authorware to work with an LMS product.

These examples are intended only to provide sample code that may be copied and modified. Some changes may be required to work with any given LMS product. There are many comments within the examples to help you better understand them.

Note: Learning Management System Knowledge Objects (LMS KOs) are available in Authorware 7. For information on using LMS KOs, see the PDF document on the Authorware CD titled *LMS Knowledge Objects for Authorware*. It can also be found on the Macromedia Authorware support site (www.macromedia.com/support/authorware).

There are four types of Authorware examples in this folder that demonstrate tracking over the web. Three of these are in the AICC folder and one is in the ADL folder.

AICC

All the AICC examples should be published to the web using the Data Tracking publishing template that is included with Authorware 7. For detailed information on AICC, visit the AICC website at www.aicc.org.

AICC/CMI

The AICC /CMI folder contains an example that uses the CMI functions that are built-in to Authorware. These functions are mapped to AICC data elements. The functions automatically detect when the file is running as a web-based or file/LAN-based piece and switch to the appropriate AICC communication means (either INI-like text files or HTTP POST commands).

The other two examples in the AICC folder both use the low-level mechanism of HTTP GET's and PUT's to send data to the server via AICC HACP and do not support LAN-based LMS products.

AICC/POST

The AICC/POST folder contains an example that uses the PostURL function. This function only appears in Authorware 4.0.3 or later. PWInt (SMI functions) for the web also requires the Authorware Web Player version 5.2 or later.

AICC/GET

The AICC/GET folder contains examples that uses ReadExtFile to generate HTTP GET requests. These examples will work with earlier versions of the Authorware Web Player and may be helpful for debugging communication with some LMS products.

ADL

The ADL sample file should be published using the template named AICC API - ADL SCORM. End-users must install this template by copying the template file AICC API - ADL SCORM.htm from the ADL/HTML folder to the publishing template folder for Authorware.

The default location for the Authorware publishing template folder is:
C:\Program Files\Macromedia\Authorware 7\HTML.

ADL/Testbed

This is a sample Authorware 7 file that uses the ReadURL function of Authorware to communicate with an ADL SCORM-compatible LMS. For example, the file uses this command to initialize communication:

```
ReadURL("javascript:API.LMSInitialize(\"\""), 30)
```

The sample file relies on the AICC API - ADL SCORM publishing template which includes JavaScript code to locate the LMS-provided API object. If you publish files intended for use with an ADL-type system and do not use this template, you will need to include your own code to locate the API object and create a handle to it named *API*.

ADL/HTML

This sample contains a small HTML file that is a publishing template to be used with the Authorware 7 one-button publishing feature. Make sure the template file AICC API - ADL SCORM.htm is copied to your local Authorware publishing template folder, as described above.

Connecting with databases

Combining the interactivity of Authorware with the power and flexibility of a database gives users a powerful tool for retrieving and managing data quickly and effectively.

Using Authorware, a company that requires employees to complete regular training courses could constantly update its records when employees complete an online course and provide an accurate list of who has completed which courses. For a retail outlet or a group of restaurant owners, Authorware connected to a database could support an electronic product kiosk that's updated daily with current information about price and product availability. A referral agency for carpenters and plumbers could use Authorware to create a simple interface to its listing of members, their specialties, and their current availability. There are many ways for Authorware and databases to work together. You'll probably think of many more as you work with them.

Authorware makes connecting with databases relatively simple by using two industry standards:

- Open Database Connectivity (ODBC), a common standard for connecting to a database
- Structured Query Language (SQL), a standard language for sending commands to and receiving information from a database

ODBC enables Authorware to communicate with a variety of commonly used databases such as Access, dBase, FoxPro, and Paradox. ODBC is designed primarily for connecting to databases located on a user's local computer or local area network (LAN). For web-packaged pieces, authors may consider using ASP or PHP scripts or MySQL facilities (www.mysql.org) to communicate to server-side databases.

To see an example of how the ODBC functions are used in an Authorware piece, select Help > Show Me and then launch "Using ODBC."

CHAPTER 10

Streaming, Intranets, and the Web

Networks—local area networks (LANs); corporate, government, and educational intranets; and the Internet—offer flexible and convenient access to information. They provide an efficient and effective way to distribute the kinds of projects you create with Macromedia Authorware, such as interactive multimedia, training applications, human resource information, kiosks, hypermedia reference titles, and online magazines.

For example, Authorware and a corporate intranet are the ideal combination for developing “distance learning” applications. Anyone who has access to the intranet—whether at a company site or by a modem—can work through a training course at any time without needing to go anywhere. If the course needs updating, you make the changes once, on the server, and everyone throughout the company has instant access to the new version.

Using a local area network is easy: Authorware treats a server attached to the network the same as the hard disk of the computer it’s running on. It also provides two elegant ways of using intranets and the Internet:

- Streaming technology, which lets you download and run Authorware pieces in Netscape Navigator, Internet Explorer, and other compatible browsers
- Internet services, which let you link to external content through uniform resource locators (URLs) and give you access to a complete set of file transfer protocol (FTP) functions even when you’re not using streaming technology

Speed

Computers keep getting faster and faster. But no matter how fast a computer can process information, the information needs to be in memory before the computer can do anything with it. Getting information to the computer and into memory over a network takes time.

There are three basic ways to use Authorware with a network, and network speed affects all of them:

- You can run an Authorware piece across a LAN.
When you launch the piece, your computer loads the Authorware runtime application into memory across the network. If the piece itself or the libraries it’s linked to are also on the server, Authorware downloads information across the network as it needs it.

- You can download a piece across the Internet or an intranet and use Macromedia Authorware Web Player to play it in a web browser.

The player downloads the segments of the web-packaged piece and any external content across the network as it needs them.

- You can run a piece that's located on your computer's hard disk but is linked to a library or external content that's elsewhere.

You can link to a library only across a LAN. You can link to external content over a LAN, over an intranet, or over the Internet.

The speed at which you can play a piece across a network, download it, or download linked external content always depends on the same things: the speed of data transmission across the network and the amount of information being transferred. On all types of networks—LANs, intranets, and the Internet—network speed varies depending on the specific network configuration and the amount of activity on the network.

Intranets that use T1 or T3 servers can be quite fast, but if users are dialing into the intranet from a modem, the connection will be only as fast as their modem.

The Internet is typically slower than an intranet, primarily because unless users have DSL or other high-speed connections, users dial in at relatively slow speeds—28,800 or 33,600 bits per second (bps). At 28,800 bps, a user receives only about 2800 data bytes per second. At 33,600 bps, the rate increases to approximately 3300 bytes per second. At those speeds, it takes 20 to 30 seconds to download a 60K file. If traffic is heavy at the Internet access point or on the Internet host, or if there's network congestion, the rate drops even lower—to as low as a few hundred bytes per second. Although 56 Kbps and faster modems are becoming more common, for now it's a good idea to assume a file will download at a rate of about 1500 bytes per second.

Streaming technology for the web

Streaming technology lets you package an Authorware piece especially for the web, download it over the Internet or an intranet, and run it in a compatible browser such as Netscape Navigator or Internet Explorer.

Streaming involves the use of two components:

- The publish for Authorware Web Player options (part of one-button publishing), which you use to prepare the piece and segment it for downloading over the web. The one-button publishing process creates a map file, which tells Authorware Web Player what to download, when to download it, and where to put the downloaded segments.
- Authorware Web Player, which manages the downloading according to the map file and runs the piece.

One of the big advantages of web-packaging a piece is that it's divided into segments. Authorware Web Player can start running the piece as soon as the first segment is downloaded—it doesn't have to wait for the entire piece. Furthermore, the player downloads segments only as they're needed. This makes it possible to produce large applications—20 or 30 MB and larger—that run efficiently and continuously. If you design a piece cleverly, users won't realize that portions of it are being downloaded as the piece is running. If you've installed Authorware Advanced Streamer on a web server, Authorware Web Player will even attempt to download in advance the segments users are most likely to want next.

Using streaming, you can deliver full-fidelity, cross-platform interactive pieces. You can deliver interactive multimedia training anywhere in your company, tracking user information, offering online help, and exchanging information between users and the file server.

You can create a central library of interactive documents for use by employees—such things as job listings, benefits information, phone directories, and performance-support aids—that can be updated at any time.

You can use virtually all the features of Authorware. You can include text, graphics, audio, animations, and movies, as well as all response types, data tracking, and database management. You can embed your piece within a web page or have it take over the full screen. Your piece can retrieve information from the intranet and send information back to the server. You can jump to other URLs and make calls to JavaScript from your piece.

The main limitation on just about everything you do is the data transmission speed of the intranet that carries your piece. The slower the transmission speed, the less ambitious you can be with using multimedia in your piece. For tips on what you can do to optimize your piece for delivery over low bandwidth, see [“Tips for optimizing web-packaged pieces for low-bandwidth \(28.8 Kbps\) delivery” on page 204](#).

Designing a piece for the web

An Authorware piece that is downloaded over the web must be designed differently from a piece that runs from the user’s hard disk. The two environments have different strengths and constraints: the major strength of streaming technology is its flexibility; its major constraint is network speed.

Orchestrating downloads

Viewers are sophisticated. To keep them engaged, you need to avoid dead spots during downloading. Design a project so that long downloads and data transfers aren’t necessary. Orchestrate the flow of information so there’s always something to engage a user’s attention during downloads and data transfers.

Here are a few suggestions:

- Keep in mind the network speed the user will experience. What may be fine over a T1 line may be unbearably slow over a 28,800 bps modem.
- Emphasize interactivity rather than big graphics and lengthy sounds. The interactivity in Authorware doesn’t require a lot of bandwidth; it works well at standard modem speeds.
- Try using a small Presentation window. It’s easier to keep the file size small if you use small bitmaps, and it’s easier to use small bitmaps if your Presentation window is small. Nonstandard Presentation windows can look great on web pages.
- Experiment with really small bitmaps. See how creative you can be with 4-bit images, which are half as big as 8-bit images. Try using 1-bit images; they’re black and white, but you can add two colors (one foreground and one background) with the Colors palette in Authorware. Line art (in which the entire image is either solid black or white) can be scanned to produce attractive 1-bit images.
- Save graphics in JPEG and GIF formats, which are smaller than their BMP counterparts. Macromedia Fireworks is an excellent tool both for determining which format produces a more compact file and for saving the file in the format that’s best.

- Save voice audio files in Voxware (VOX) format. Save other audio files in Shockwave Audio (SWA) or MP3 format.
- Make the most of the `NetDownload`, `NetPreload`, and `Preload` functions. (See [“Web-related variables and functions” on page 206.](#))
- When you use one-button publishing to package a piece for Authorware Web Player, experiment with different size segments.
- Consider the possibility of distributing large media files on CD. (See [“CD–web hybrids” on page 212.](#))
- Test your work at the speed the user might experience. Ask yourself if the user will be willing to wait for the file to download.

For suggestions on improving performance, see Performance tips for web-packaged pieces and Tips for optimizing web-packaged pieces for low-bandwidth (28.8 Kbps) delivery in Authorware help.

Tips for optimizing web-packaged pieces for low-bandwidth (28.8 Kbps) delivery

The following tips are useful for delivering pieces over low bandwidth.

- For voice-overs and narration that is “clean voice” with no echoes or background noise or music, use VOX compression. It will produce very small files.
- Use the Export Media command to export internal sounds and images to compress them into a smaller size (for example, SWA, MP3, VOX, and GIF). This is particularly useful if you no longer have the original source material for the sounds and images that are now internal to the piece.
- Use internal media whenever possible because it works better with the Advanced Streamer. External media (including references to URL-based media) are not predictively downloaded with the Knowledge Stream technology. You can use the Package External Media Internally option in one-button publishing to facilitate this.
- Avoid using AVI or QuickTime movies at 28.8 Kbps. If you must use a movie, use a tool such as Adobe Premiere to save it as an FLC/FLI and import it into Authorware as an internal movie, and use a VOX, SWA, or MP3 file in a sound icon as the soundtrack.
- Avoid using large external U32 libraries and custom Xtra extensions if you can. If you do use these, use recycle so it doesn’t have to be downloaded each time. U32 and Xtra extensions are not part of what the Knowledge Stream technology knows about and are not predictively downloaded.
- Use a single replicated tiled graphic (stored in a library) instead of a large screen.
- In general, store graphics that are shared among pieces in external libraries.
- A single large AAM file (that is, from a single A7P file) works better than multiple AAM files. If you don’t share libraries between multiple AAM files, package the libraries internally.
- If you are having bandwidth problems, first try compressing the graphic or sound being displayed just after the hiccup. If that doesn’t work, try inserting wait icons at opportune places in the course to get delays at predictable times rather than randomly. If you insert a wait icon, the Authorware Web Player will continue preloading (predictively) the next segments to load and will “get ahead” so that it can overcome any temporary higher-bandwidth parts of the course.
- For computer-generated animation style graphics, use RLE or GIF.

- For natural photographic style images use JPEG. Don't use internal bitmaps.
- Put in "Loading..." animation at the very beginning of the piece, followed by a wait icon. This will give the first parts of the course a chance to preload so they flow smoothly and the user gets feedback that the piece is still running.
- To compress graphics, use File > Export Media to export them and then import them into Macromedia Fireworks, or another program such as Adobe Photoshop or Debabelizer. Fireworks will let you try out different compression formats (for example, GIF versus JPEG) side-by-side to see which produces better results. Then re-import them into Authorware.
- Export sounds, if necessary, by selecting File > Export Media. Then use the SWA Xtra to convert these to SWA. Use 8 Kbits/high compression. Record the WAV files at the best sample rate you can (but usually 22 kHz/16-bit will suffice). The better the input, the better sounding the compressed sounds will be, and the size of the SWA files doesn't depend on the size of the input files. For voice sounds which have no background music or noise and no echo (for example, just "clean" speech), use the Voxware Encoder that Authorware installs in your Windows Start program menu.
- Set the calculation variable `GlobalPreroll` to 15000 bytes or so. You can also change the preroll on a per-sound-icon basis. Prerolls make sure that sound is loaded before it is needed and acts as a buffer to allow for hiccups in network bandwidth. The sound won't start playing until the preroll number of bytes is downloaded.
- Put a wait icon at the beginning of your piece to give the Advanced Streamer a chance to act.
- If you are using `NetDownload()`, consider converting it to `NetDownloadBackground()` to make the download happen in the background.
- To see if your piece is approximately the right size to work at a speed over 28.8 Kbps, web-package it to an empty folder and use Windows File Properties to see the overall size of the folder contents. Estimate how long you expect your course to take in real time (that is, does it encompass five minutes of online training of a student sitting in front of the computer). Take the overall size of the web-packaged piece (such as 500K) and divide by the real time length of the piece (such as 5 minutes). For 28.8, realistically you get about 2K/sec or 120K/minute bandwidth. For this example, 500K/5 minutes is 100K/minute, which should work well at 28.8K. If your speed is more than 120K/minute you may have trouble.
- You don't have to convert every sound and graphic to a compressed format, but graphics larger than about 8K may cause stuttering in the playback (remember you get about 2K/second at 28.8 modem speed).
- Create an icon size report to find bandwidth hogging icons.
- Reduce the bit-depth of graphics.
- Decompose large area graphics into smaller pieces if the area can be reduced (for example, break a picture frame down into four separate pieces—left, top, right, bottom—when the center portion is not important). Macromedia Fireworks can do much of this work for you automatically.
- If multiple graphics share some portion of an image in common, break the images into component pieces and place them in a library. Build the full graphic using display links to the component pieces. (For example, a software training piece may have several screenshots where the contents of the file windows differ, but the toolbar remains the same.)
- If a background is repeating, place one small tile in a library, and build up the background with multiple display icon links to the tile.
- Use built-in toolbox patterns instead of large complex image backgrounds.

- Break down video into narration (sound icon) and flip through a few static frames (using transitions to spice it up).
- Use native toolbox objects instead of images whenever possible (for example, a text object instead of an image containing text).
- Avoid using `Preload()` because it blocks (it doesn't work in the background). The Knowledge Stream technology does much of this preloading. If you have to, use `NetPreload()` instead, but use it sparingly because it will do manually what the predictive downloading tries to do automatically.

Web-related variables and functions

A special set of variables and functions has been designed for use with web-packaged Authorware pieces. They appear in the Network category in the Functions dialog box.

Variables

Variable	Description
<code>GlobalPreroll</code>	Contains the number of bytes to download before any network-based sounds used by your piece begin to play.
<code>NetBrowserName</code>	Contains the name of the browser running the current piece.
<code>NetBrowserVendor</code>	Contains the name of the vendor of the browser running the current piece.
<code>NetBrowserVersion</code>	Contains the version number of the browser running the current piece.
<code>NetConnected</code>	Indicates whether Authorware Web Player is running a piece. If you create two different versions of the same piece—one web-packaged, one not—you can use <code>NetConnected</code> to determine which is running. You might use that information to display a 1-bit line drawing for the web-packaged version and an 8-bit image for the other.
<code>NetLocation</code>	Contains the server location where the web-packaged piece is located. With <code>NetLocation</code> you don't have to keep track of the URL yourself. It's an easy way to find other things, such as external content files, stored at the same location.
<code>Preroll</code>	Contains the number of bytes to download before the sound contained in the specified icon begins to play.
<code>SoundBytes</code>	Contains the size (number of bytes) of the sound contained in the specified sound icon.

Functions

Function	Description
<code>Bandwidth</code>	Returns the rate of previous reads or downloads in bytes per second.
FTP U32 functions	Add FTP capabilities to Authorware. See “The Authorware FTP functions” on page 211 .
<code>JumpFile</code>	Jumps to a URL-specified Authorware piece. (URLs work only in web-packaged pieces.)

Function	Description
JumpFileReturn	Jumps to a URL-specified Authorware piece and then returns. (URLs work only in web-packaged pieces.)
GoToNetPage	Opens a new web page identified by a URL in the browser. Use <code>GoToNetPage</code> to go from a web-packaged piece either to a web page or to another web-packaged piece.
NetAbort	Attempts to terminate a download given a NetID returned by <code>NetDownloadBackground</code> .
NetDownload	Downloads a file identified by a URL to the user's hard disk. Use <code>NetDownload</code> to download content, such as text or graphics, or files Authorware Web Player may need, such as sprite or scripting Xtra extensions.
NetDownloadBackground	Attempts a background download of the file specified in URL to the given local filename. <code>NetDownloadBackground</code> is similar to <code>NetDownload</code> , but doesn't wait for a download to complete.
NetDownloadName	Returns a destination filename given a netID as returned by <code>NetDownloadBackground</code> .
NetError	Returns an error code for a download given a NetID as returned by <code>NetDownloadBackground</code> .
NetFileSize	Returns the content size in bytes of the specified URL.
NetLastModDate	Returns the most recent modification date of the specified URL.
NetPercentDone	Reports the progress of a download given a NetID as returned by <code>NetDownloadBackground</code> . Progress ranges from 0 to 100.
NetPreload	Downloads a segment of a web-packaged piece. Use <code>NetPreload</code> to download segments when it will be least noticeable to the user; don't use <code>NetPreload</code> , however, if you're using Authorware Advanced Streamer. To preload external content linked to a piece, use the <code>Preload</code> function.
ReadExtFile	Reads an external file identified by a URL.
ReadURL	Reads the specified URL and returns the file's contents, or the javascript result, as a character string.
ResetBandwidth	Clears the bandwidth statistics for the specified selector.

If you use the `GoToNetPage` and `NetDownload` functions in an Authorware piece that hasn't been packaged for the web, Authorware ignores them.

If you use `NetPreload` in a piece that hasn't been packaged for the web, it works just like `Preload`. The `Preload` and the `NetPreload` functions are related but do different things:

`Preload` loads information into memory so that it's immediately accessible when Authorware is ready to use it. `Preload` is usually used with large files, such as embedded digital movies, which can produce a noticeable lag if Authorware needs to load them into memory before it can run them. You can download external graphics and sounds directly into memory by using a URL with `Preload`; this works only with external graphics and sound.

`NetPreload` downloads a segment of a web-packaged piece to the user's computer (it won't, however, download external sounds or graphics files). Use it to download a segment behind the scenes so it's at hand when Authorware needs it. Unlike `Preload`, `NetPreload` doesn't load the segment into memory.

Managing content in web-packaged pieces

Content can be embedded in or external to a web-packaged piece, just as it can in any Authorware piece. A consideration to keep in mind when designing a piece is to avoid delays when content is downloading.

If you're using Authorware Advanced Streamer, avoid using external content whenever possible. It's less efficient than embedded content.

Embedded content Authorware one-button publishing includes embedded content in one of the segments that it creates when you package the piece for the web.

- If you're using Authorware Advanced Streamer, avoid using the `NetPreload` function. `NetPreload` does manually what Authorware Advanced Streamer attempts to do automatically: download segments when the transfer is least noticeable.
- If you're not using Authorware Advanced Streamer, use the `onDemand` option and `NetPreload` to control when Authorware Web Player downloads the segments that contain content.

External content Authorware handles external content differently depending on how you've set up the content:

- If the path to an external media file points to your local hard disk or your local server, one-button publishing adds a path that points to the folder on the user's computer where the player will download the file.
Use the `onDemand` option to download the file only if and when it's actually needed: the player first checks for the file in the folder on the user's computer; if it finds a copy there, it uses it rather than downloading the file again.
- If the path to an external media file is a URL, one-button publishing leaves the path as it is.
Use `Preload` to load an external sound or graphics file directly from a URL into memory.
- If you use `NetDownload` to download an external media file, Authorware Web Packager leaves the path to the file as it is.
Use the `onDemand` option in the map file to control precisely when the player downloads the file.

Note: Custom fonts are difficult to handle because they need to be installed after they're downloaded (see Using custom fonts in web-packaged pieces).

Configuring your server

You need to set up the web server where you place the HTML page and the web-packaged Authorware piece so that it recognizes the Multipurpose Internet Mail Extensions (MIME) type that identifies a web-packaged Authorware file. It's a matter of adding a few lines to the server's configuration settings.

Providing the Authorware Web Player

Users need the correct version of Authorware Web Player to run a web-packaged piece you've created in Authorware 7.

If you aren't working in an environment where you can distribute the player to users directly, have them download it from the Macromedia website (it's free). For information about setting up a link to the Macromedia website, visit the Authorware Support Center. (Select Help > Support Center or visit www.macromedia.com/support/authorware.)

Streaming-step-by-step procedures

What you can do	In Authorware Help, see
Package a piece for the web	Getting a piece ready for the web, Publishing for the Authorware Web Player, Publishing to a web page, Using one-button publishing to package a piece
Edit a map file	Editing a map file with the Authorware Web Packager editor
Combine map files for different player versions	Combining map files for different player versions
Add a line to a map file	Adding a new external content line
Edit map segment lines	Adding a new external content line
Edit library lines	Adding a new external content line
Add a line for external content	Adding a new external content line
Edit an external content line	Adding a new external content line
Use Xtra extensions in web-packaged pieces	Web-packaged pieces and external files
Play a Director movie in a web-packaged piece	Playing Director movies in web-packaged pieces
Use multiple get and put entries	Using multiple get and put entries
Tune the performance of a web-packaged piece	Performance tips for web-packaged pieces
Bypass the Security dialog box	Bypassing the Security dialog box in trusting mode, Bypassing the Security dialog box in nontrusting mode
Check whether a user is running in trusting or nontrusting mode	Querying a network security variable
Create a piece that won't be affected by any browser's palette	Creating a web-packaged piece with a web-safe palette
Configure the web server	Configuring your server
Add a web-packaged piece to a web page	Publishing to a web page, Using one-button publishing to package a piece
Set up a web page for non-streaming browsers	Working with non-streaming browsers

Authorware Advanced Streamer

Authorware Advanced Streamer improves downloading performance for a web-packaged piece. Authorware Advanced Streamer records the general file access patterns that users make when running a piece. It uses that information to predictively download file segments in the background when subsequent users run the piece.

Authorware Advanced Streamer uses probability calculations to determine how future users will navigate through the piece based on past users. As a user spends time reading information or interacting with the current segment of a piece, Authorware Advanced Streamer downloads the next segments that it predicts will be requested by the user. By productively using this idle time, Authorware Advanced Streamer reduces the amount of time users spend waiting for material to download.

For detailed information about Authorware Advanced Streamer, see Authorware Help.

Using Internet services

Your use of the web isn't limited to what you can do in web-packaged pieces. In standard Authorware pieces—and without a web browser—you can:

- Link to external content files on a web server
For example, on a corporate intranet you could provide brief QuickTime reports on company events and update them daily.
- Use the FTP functions in the Authorware FTP U32.
Whenever a user completes a section of a training course, you can use the `FTPStore` function to transfer the results to a central web server.
- Use `ReadURL` to call JavaScript.

You can also use the Macromedia Control Xtra for ActiveX to embed Internet Explorer in an Authorware piece and use it to display HTML pages or to connect with the web. For example, if you're developing a training piece on the Internet, you could launch Internet Explorer and respond to actions users take in the browser.

Designing for Internet services

Many of the issues you have to deal with when you're creating a web-packaged piece also apply to a standard Authorware piece that uses Internet services:

- Keep in mind the network speed the user will experience.
- Avoid delays during downloading.
- Link to external content over the web only when you must, and then keep graphics, sounds, and movies as small and efficient as possible.
- Remember that your content needs to be compatible with the computer it's running on.
- Make the most of the FTP functions that Authorware can use (see [“The Authorware FTP functions” on page 211](#)).
- Test your work at the speed the user will experience.

There are differences too. Some variables, functions, and features are available only to web-packaged pieces:

- Of the Authorware web-related variables, only `GlobalPreroll`, `Preroll`, and `SoundBytes` can be used in both standard and web-packaged pieces. `NetConnected` and `NetLocation` work only in connection with web-packaged pieces. `NetBrowserName`, `NetBrowserVendor`, and `NetBrowserVersion` work only in web-packed pieces.
- Of the Authorware web-related functions, only `Bandwidth`, `Jumpfile`, `JumpFileReturn`, `ReadExtFile`, and `ResetBandwidth` can be used in both standard and web-packaged pieces. (For a complete list of web-related functions, see [“Web-related variables and functions” on page 206.](#))
- You can use URLs with `JumpFile` and `JumpFileReturn` only in web-packaged pieces. So you can't jump from a local Authorware file to an Authorware file on a web server.
- You can use the `Preload` function to preload an internal sound or graphic or an external sound or graphic linked to a URL into memory.
- With a standard Authorware piece, there are no download options that let you download on demand or preempt or recycle a file that's already been downloaded.
- The Internet services available to a standard piece have no built-in security features. Xtra extensions, U32s, and DLLs can be downloaded freely; the FTP functions can read and write files.

The Authorware FTP functions

The Authorware FTP U32 contains 19 functions that you can use to manage the transfer of information over the Internet or an intranet. The following table gives a brief description of each function. For more information, see the Read Me file that accompanies the U32.

Function	What it does
<code>FtpAbort</code>	Aborts the current asynchronous function.
<code>FtpAppend</code>	Transfers a file from the local file system to a web server. If the file doesn't exist, <code>FtpAppend</code> creates it. If the file does exist, <code>FtpAppend</code> appends the transferred file to the existing file.
<code>FtpChangeParentDir</code>	Simplifies dealing with different server syntaxes for naming the parent directory (this is a special case of <code>FtpChangeWorkingDir</code>).
<code>FtpChangeWorkingDir</code>	Allows the calling application to use a different directory for file storage or retrieval.
<code>FtpClose</code>	Closes the FTP session.
<code>FtpConnect</code>	Establishes an FTP connection to a server.
<code>FtpDelete</code>	Deletes a file from a web server.
<code>FtpDisconnect</code>	Disconnects the FTP connection from the server.
<code>FtpGetWorkingDir</code>	Gets the full path of the current working directory.
<code>FtpList</code>	Instructs a web server to send a list of files.
<code>FtpMakeDir</code>	Causes the directory specified by the remote path to be created as a directory.

Function	What it does
FtpNameList	Instructs a web server to send a directory listing.
FtpOpen	Allocates and initializes resources for the FTP session.
FtpRemoveDir	Causes the directory specified by the remote path to be removed as a directory as long as it's empty.
FtpRename	Renames a file on a web server.
FtpResult	Returns the result string of the completed FTP operation.
FtpRetrieve	Retrieves a file from a web server.
FtpStatus	Returns the status of the last function.
FtpStore	Stores a file on a web server.

CD-web hybrids

The main constraint on delivering multimedia over a network is always network speed. Productions with a lot of audio, video, and graphics are especially restricted because these types of media take up the most disk space. CD-web hybrids overcome this limitation by combining the media-rich content of CD technology with the immediacy and resources of the Internet.

For example, a CD-based travel guide could download the most up-to-date information about hotel prices, entertainment, and exchange rates from the Internet. You could either package such a piece for the web and run it in a browser from the CD—at about 100 times the speed of downloading it from the web over a 28,800 bps modem—or you could package it and use the Authorware Internet services to download information from the web.

Whichever method you select, the combination of a CD and the web can be an ideal solution for many projects.

CHAPTER 11

Creating Knowledge Objects

A Knowledge Object is a packet of logic—encapsulated in a model—that you insert into a Macromedia Authorware piece. A Knowledge Object is different from a simple model in that it's connected to a wizard. A wizard is an Authorware piece that provides an interface for setting up, making changes to, or even adding new content and logic to the Authorware piece where you've inserted the Knowledge Object.

Knowledge Objects are powerful productivity tools that make inexperienced members of a development team productive by enabling them to do common tasks easily and efficiently. Knowledge Objects also make you more productive if you're an experienced developer by giving you a way to automate repetitive tasks. Knowledge Objects, in fact, fundamentally revolutionize the authoring practice.

The following are examples of some of the things you can do with Knowledge Objects:

- You're creating a psychological evaluation with hundreds of multiple-choice questions. After the content of the questions has been developed by experts, you create a Knowledge Object that allows a low-level Authorware specialist to import each question from a text file and set up an interaction for it—without using any of the detailed Authorware dialog boxes.
- You recognize that there's a need for templates that help trainers construct courses that embody good instructional design principles. You create Knowledge Objects that embody such templates. You package and sell them.
- You're creating a huge project in Authorware that has thousands of framework pages. You discover well into the project that you need to add a prefix to the name of every page. You can create a Knowledge Object that changes the title of every page icon automatically.
- You discover that you need to create a set of Knowledge Objects that all do very much the same thing. You can create a Knowledge Object that helps you build the other Knowledge Objects.
- You're a consultant, and after you deliver a project you've created in Authorware to your client, the client discovers a few minor bugs in it. You can create a Knowledge Object the client can run to fix the bugs.

A Knowledge Object works like this: A user drags a Knowledge Object from the Knowledge Objects window to the flowline. The Knowledge Object launches the wizard connected to it. Doing so launches another instance of Authorware in which the wizard runs. The wizard leads the user through the process of setting up content in the Knowledge Object or configuring the code it contains. As the user makes choices in the wizard, the wizard makes the appropriate changes in the “target” Authorware piece (the piece where the user has inserted the Knowledge Object). When the user quits the wizard and runs the file that contains the Knowledge Object, Authorware, instead of launching the wizard when it encounters the Knowledge Object icon, executes the logic inside the icon.

The simplest Knowledge Objects consist of a Knowledge Object icon and the wizard it launches. You save a Knowledge Object icon as a model. Knowledge Objects—as long as you save them in the proper folder—appear in the Knowledge Objects window, where they’re easily accessible to users. They can be used over and over again.

Displaying Knowledge Objects in the Knowledge Objects window

The Knowledge Objects window provides access to the Knowledge Objects and models that are located in the Knowledge Objects folder in the Authorware 7 folder.

The Category drop-down list at the top of the Knowledge Objects window lists the name of each of the subfolders inside the Knowledge Objects folder. “All” also appears on the drop-down list. Choosing All lists all the Knowledge Objects and models in all of the subfolders and any Knowledge Objects and models in the Knowledge Objects folder itself. Selecting any other category lists the Knowledge Objects and models in the subfolder it represents. Knowledge Objects and models in subfolders nested inside subfolders aren’t listed in the Knowledge Objects window.

The Category drop-down list shows shortcuts to folders outside the Knowledge Objects folder; likewise, the Knowledge Objects window shows shortcuts to Knowledge Objects that are stored in folders outside the Knowledge Objects folder (as long as the shortcuts are in the Knowledge Objects folder or one of its subfolders). Shortcuts appear in italics.

If you add Knowledge Objects to the Knowledge Objects folder or any of its subfolders after you’ve launched Authorware, click the Refresh button in the Knowledge Objects window.

Note: Restricted users on Windows systems may not be able to install Knowledge Objects in the Knowledge Objects folder.

Wizards and targets

A wizard and its target are two separate Authorware pieces in two separate instances of Authorware that are running simultaneously. The wizard is an interface for setting up content or for configuring or constructing logic in the target Authorware piece. It contains logic that manipulates the target Authorware piece.

Launching a wizard

When you launch a wizard from a Knowledge Object, it automatically starts another instance of Authorware. The instance of Authorware running the wizard takes focus so that it's impossible for a user to make changes in the target piece while the wizard is running, except through the wizard. A wizard can make changes not just inside the Knowledge Object that launched it but to any part of the target Authorware piece.

A wizard communicates with the target Authorware piece through the Knowledge Object that launched it. A wizard can, of course, be launched by different instances of the same Knowledge Object in the same Authorware piece or by the same Knowledge Object in different Authorware pieces—but only one wizard can be running in an Authorware piece at a time.

When you pause an unpackaged wizard (which you may do when you're testing or debugging a wizard), you can make changes to the wizard and then resume running it or restart it from the beginning.

Quitting a wizard

When you quit a packaged wizard, the instance of Authorware running it quits automatically.

When you quit an unpackaged wizard, the wizard file closes, but the instance of Authorware running it remains open. You can either close it or switch focus to return to the instance of Authorware containing the target file.

Variables and functions for Knowledge Objects

The variables and functions for Authorware Knowledge Objects are described in the following sections.

You can use all of these functions and variables from a wizard, just as they are used in commands. However, if you open a new file, you will lose the Knowledge Object icon reference point into the piece. You'll need to reference icons in the target through some other means, such as the `RootIcon` function. Also, if you package the piece, the icon IDs in the target piece may change in the process.

Variables

For detailed information about the following variables, see *System Variables in Authorware Help*.

Variable	Description
<code>KnowledgeObjectID</code>	<code>KnowledgeObjectID@"IconTitle"</code> contains the Knowledge Object ID string from the icon specified in <code>IconTitle</code> .
<code>TargetIcon</code>	<code>TargetIcon</code> contains the icon ID of the Knowledge Object icon from the target piece responsible for launching the current wizard file.

Functions

For detailed information about the following functions, see [System Functions in Authorware Help](#).

Function	Description
CallTarget	Executes a function in a target piece. Use CallTarget only in calculation icons in a wizard (see “Communicating with a Knowledge Object” on page 221).
ClearIcons	Deletes the selected icons without copying them to the Clipboard.
CopyIcons	Copies the selected icons to the Clipboard.
CutIcons	Cuts the selected icons and places them on the Clipboard.
DisplayResponse	Displays the response associated with the specified icon.
EraseResponse	Erases the response associated with the specified icon.
GetCalc	Returns the contents of the specified calculation icon (or the calculation attached to any specified icon) as a string.
GetExternalMedia	Returns a full list of external media associated with the node or the library. This function returns an empty list when used on packaged files.
GetFileProperty	Gets the file property for the indicated property. This function only operates on unpackaged files. Supported properties include: #awTitleBar, #awTaskBar, #awCenterOnScreen, #awMenuBar, #awOverlayMenu, #awMatchWindowColor, #awStandardAppearance, #awWindows31Metrics, #awDesktopPattern, #awTrackAllInteractions, #awTrackScore, #awTrackTime, #awTrackTimeout, #awLogoutUponExit.
GetFunctionList	Returns a property list with information about the functions within the specified category. Use 0 to retrieve the list of functions in U32 and Script Xtra extensions loaded by your course. This function only operates on unpackaged files.
GetIconContents	Returns a property list of content contained within the specified icon.
GetInitialValue	Retrieves the initial value of a variable.
GetLibraryInfo	Returns a linear list of all libraries currently associated with the current course file.
GetPasteHand	Returns the IconID of the icon closest to the paste hand. If the IconID is positive, then the paste hand is before that icon. If the iconID is negative, then the paste hand is after that icon. If the value returned is 0, then the paste hand is on the Tool palette. This function only operates on unpackaged files.
GetPostPoint	Returns the post point for the icon you specify. The post point is the upper left corner of the bounding box for objects in a display or response. The return value looks like this: point(x, y).
GetPostSize	Returns the post size for the icon you specify. The size is returned like this: size(width, height).
GetSelectedIcons	Returns a linear list describing the selected icons in the front map. Values are the IconTitle, IconID, and IconType. This function only operates on unpackaged files.

Function	Description
<code>GetVariable</code>	Retrieves the value of the named variable.
<code>IconID</code>	Returns the unique numeric identifier of the icon in which it appears. Use <code>IconID("IconTitle")</code> to get the ID of a specific icon.
<code>ImportMedia</code>	Imports media into the specified display, interaction, movie, or sound icon.
<code>InsertIcon</code>	Inserts an icon of the specified <code>IconType</code> on the Authorware flowline at the current location of the Paste Hand.
<code>IsCourseChanged</code>	Returns <code>true</code> if the Course files has been changed and not saved. This function only operates on unpackaged files.
<code>IsLibraryChanged</code>	Returns <code>true</code> if the Content Library file specified by <code>LibraryID</code> has been changed and not saved. Use <code>GetLibraryInfo</code> to obtain a list of libraries containing the ID for each file. This function only operates on unpackaged files.
<code>KORefresh</code>	Reloads the Knowledge Objects window from the Knowledge Objects folder on your hard drive. From within a script, use to reread the Knowledge Objects folder and refresh the Knowledge Objects dialog box.
<code>NewVariable</code>	Creates a new user variable. Use an <code>IconID</code> to create a local variable. Sets <code>EvalStatus</code> . This function only operates on unpackaged files.
<code>OpenFile</code>	Opens the specified course file. If the filename is omitted, a new untitled file is created.
<code>OpenIcon</code>	Opens the specified icon
<code>OpenLibrary</code>	Opens the specified content library file. Returns the value of <code>IOStatus</code> . This function only operates on unpackaged files.
<code>PackageFile</code>	Packages the currently open course file. Returns the value of <code>IOStatus</code> . This function does nothing unless called via the <code>CallTarget</code> function on an unpackaged target file.
<code>PackageLibrary</code>	Packages the specified library file. The file specified must be connected with the current course. Returns the value of <code>IOStatus</code> . This function does nothing unless called via the <code>CallTarget</code> function on an unpackaged target file.
<code>PasteIcons</code>	Pastes icons from the Clipboard to the point on the flowline indicated by the Paste Hand.
<code>PasteModel</code>	Inserts the specified model into a piece at the point on the flowline indicated the Paste Hand.
<code>SaveFile</code>	Saves the current course file. Untitled files are not saved if the filename is omitted or the path is relative. Sets <code>IOStatus</code> and <code>IOMessage</code> . Returns the value of <code>IOStatus</code> . This function only operates on unpackaged files.
<code>SaveLibrary</code>	Saves the specified content library file. If a new filename is given, the library is saved to the new file and the original file is left intact. The course file will be marked as "Changed" during this action. Returns the value of <code>IOStatus</code> . This function only operates on unpackaged files.
<code>SelectIcon</code>	Clears, sets, or adds to the selection in the piece.

Function	Description
SetCalc	Sets the calc of the specified icon, if the calculation is valid. Passing <code>true</code> for language sets the calculation to JavaScript mode. Passing <code>false</code> or omitting language sets the calculation to Authorware script. Sets <code>EvalStatus</code> after compiling expression.
SetEmpty	Sets the state of the Empty property of the specified Knowledge Object icon.
SetFileProperty	Sets the file property for the indicated property to the indicated value. This function only operates on unpackaged files. Returns <code>true</code> if successful, <code>false</code> otherwise. Supported properties include: #awTitleBar, #awTaskBar, #awCenterOnScreen, #awMenuBar, #awOverlayMenu, #awMatchWindowColor, #awStandardAppearance, #awWindows31Metrics, #awDesktopPattern, #awTrackAllInteractions, #awTrackScore, #awTrackTime, #awTrackTimeout, #awLogoutUponExit.
SetIconTitle	Sets the title of the specified icon.
SetInitialValue	Sets the initial value of the named variable.
SetPasteHand	Sets the location of the Paste Hand.
SetPostPoint	Sets the post point for the icon you specify. The post point is the upper left corner of the bounding box for objects in a display or response.
SetPostSize	Sets the post size for the icon you specify.
SetVariable	Sets the value of the named variable.
ShowWindow	Opens or closes the Presentation window.

Knowledge Objects—step-by-step procedures

What you can do	In Authorware Help, see
Create a Knowledge Object and the wizard that provides its interface	Creating a Knowledge Object, Creating a good wizard interface
Add a Knowledge Object icon to the flowline and set it up	The Knowledge Object icon, Setting up a Knowledge Object icon
Connect a Knowledge Object icon and the wizard that provides its interface	Connecting a Knowledge Object icon to a wizard
Create a unique ID for a Knowledge Object icon	Creating a unique ID for a Knowledge Object icon
Use the CallTarget function to execute a function in a wizard	CallTarget function, Communicating with a Knowledge Object
Get and set variable values with a wizard	GetCalc function, SetCalc function, and Getting and setting variables with a wizard
Use a wizard to position the Paste Hand in the target piece	SetPasteHand function, Positioning the Paste Hand with a wizard
Use a wizard to select icons in the target piece	SelectIcon function, Selecting an icon with a wizard
Use a wizard to insert and delete icons in the target piece	CutIcons function, CopyIcons function, PasteIcons function, ClearIcons function, Inserting and deleting icons with a wizard, Building logic on the fly with models
Use a wizard to insert a model in the target piece	PasteModel function, Inserting and deleting icons with a wizard
Use a wizard to get and set icon properties in the target piece	Getting and setting icon properties with a wizard
Use a wizard to get and set the string in a calculation in the target piece	GetCalc function, SetCalc function, Setting a calculation with a wizard
Use a wizard to import images, movies, and sounds into the target piece	ImportMedia function, Importing media with a wizard
Give users the ability to change the settings they've selected in the wizard	Making Knowledge Objects re-entrant
Anticipate error conditions	Handling error conditions in Knowledge Objects
Test and debug a Knowledge Object	Testing and debugging Knowledge Objects
Create a preferences file for a Knowledge Object	Creating a preferences file for a Knowledge Object
Turn a Knowledge Object icon into a Knowledge Object	Turning a Knowledge Object icon into a Knowledge Object
Distribute Knowledge Objects	Distributing Knowledge Objects
Make the Knowledge Object accessible from the Knowledge Objects window	Displaying Knowledge Objects in the Knowledge Objects window

The Knowledge Object icon

The Knowledge Object icon now appears in the Icon palette. It's also accessible by selecting Insert > Icon > Knowledge Object Icon.

You can think of a Knowledge Object icon as a map icon that has a communication link with another Authorware piece running in a separate instance of Authorware. The Knowledge Object icon launches the wizard and provides the link between the wizard and the target Authorware piece where the icon is located.

Knowledge Object icons, unlike other Authorware icons, behave differently depending on whether or not you've saved them in a model:

- You can lock a Knowledge Object icon so that after you've saved it in a model, no one can open the icon and cut or copy its contents.
- You can set up a Knowledge Object icon so that after you've saved it in a model, when users drag the model from the Knowledge Objects window to the flowline, Authorware launches the Knowledge Object's wizard automatically.

Knowledge Object icon options

Every Knowledge Object icon has three options:

Lock Icon: When Saved in Model has two benefits. It protects proprietary code from unauthorized use. It also prevents people from unintentionally making changes to the logic in the Knowledge Object icon that might cause the Knowledge Object to stop working properly.

Run Wizard: Immediately When Inserted From Model launches the wizard associated with a Knowledge Object automatically when a user drags the Knowledge Object from the Knowledge Objects window to the flowline.

Empty: Run Wizard When Icon Encountered launches the wizard associated with a Knowledge Object when the Knowledge Object icon is encountered on the flowline. When a wizard needs information from the user, set the Empty property with the `SetEmpty` function. Make sure you clear the Empty property once your wizard has the information it needs. If you don't clear this setting, the Knowledge Object will launch the wizard every time the user tries to run the piece. When in doubt, clear this setting.

Creating a good wizard user interface

How good a Knowledge Object is depends not only on how well coded and well tested it is but on how successful users are in getting it to do what they want it to do. That requires a well-designed user interface.

A good design provides:

- A map. Users should be able to tell from the interface what functions are part of the wizard and where they're located in the wizard.
- Clear choices. Users should understand what they're being asked and should have enough information to make the choices they want. Provide defaults that are logical and intuitive.
- Predictable outcomes. Users need to understand the implications of the choices they make so they get the results they expect.

- Context. If a wizard completes only part of a larger task, the wizard should make clear how its function fits into the larger task.
- Re-entrancy. Users should be able to run a wizard more than once to modify their work. They shouldn't have to delete their work and run the wizard again to change the results.

Communicating with a Knowledge Object

Use the `CallTarget` function to control the target Authorware piece from the wizard. The `CallTarget` function lets you call a function and have it affect the target piece that contains the Knowledge Object that launched the wizard.

Here are some of the things you can do with the `CallTarget` function:

- Get the ID of an icon in the target piece:
`BackgroundID := CallTarget("IconID", "Background")`
- Select an icon in the target piece:
`CallTarget("SelectIcon", BackgroundID)`
- Copy the selected icon or icons in the target piece:
`CallTarget("CopyIcons")`
- Set the position of the Paste Hand in the target piece:
`IntroTextID := CallTarget("IconID", "Intro Text")`
`CallTarget("SetPasteHand", IntroTextID, #before)`
- Paste the contents of the Clipboard at the place marked by the Paste Hand:
`CallTarget("PasteIcons")`
- Change the title of an icon:
`TextResponseID := CallTarget("IconID", "Text Response")`
`CallTarget("SetIconTitle", TextResponseID, "Marie Curie")`

CHAPTER 12

Show Me Examples

Each Show Me example is a capsule computer-based training (CBT) course that explains and demonstrates a valuable technique for Macromedia Authorware. All the Show Me examples contain flowline logic that you can paste into your course to give you a head start with your work.

Accessing Show Me examples

You can access Show Me examples in two ways:

- You can open the Show Me folder in the Authorware program folder on the Authorware CD and double-click on a Show Me A7P file to open it in Authorware.
- You can go to the Authorware support site at www.macromedia.com/support/authorware/, where new Show Me examples are continually added.

24 Points (24points.a7p)

Uses the Authorware target area interaction to create a drag-and-drop math game.

Administration (Admin.a7p)

Provides a simple application to read the data written by the Logon and Content Show Me files.

Altimeter (Altimeter.a7p)

Simulates an altimeter by using the Line function and the capability of Authorware to update displayed variables.

Animated Buttons (anibutt.a7p)

Demonstrates how to set up an animated button.

A/V Synchronization (Avsync.a7p)

Illustrates synchronizing text or graphics with audio and video.

Bar Graph (Bargraph.a7p)

Demonstrates how to create a simple bar graph.

Button Interaction (Q_button.a7p)

Provides a simple demonstration of a push button interaction.

Calculator (Calculator.a7p)

Demonstrates how to build a simple calculator.

Catch It (CatchIt.a7p)

Demonstrates how to use the Authorware `Overlapping` function to build a simple game called Catch It.

CD-ROM Drive Letter (FindCD.a7p)

Demonstrates how to determine the drive letter assigned to the CD-ROM drive.

Clock (Clock.a7p)

Demonstrates how to create an analog clock using the Authorware `Line` function.

Content Application (Content.a7p)

Provides a simple application that demonstrates how to integrate smaller pieces into a complete application.

Creating a Subroutine (Subrout.a7p)

Shows how to create a subroutine and some uses for subroutines, such as recording student scores.

Cubes (Cubes.a7p)

Show how to represent 3D cubes using the Authorware line tool.

Displaying Variables (Variable.a7p)

Provides a simple demonstration of variables embedded in a display. Shows how to use the `Update Displayed Variables` setting to display the current icon title, time, and other user information.

Double Click Selection (Dblickclick.a7p)

Shows how to set up double-click and single-click responses in one interaction.

Drag and Drop Interactions (Drag.a7p)

Demonstrates how to use the Authorware system functions and variables to simplify the setup for target area interactions.

Encryption-Decryption (Encrypt.a7p)

Demonstrates how to build a simple encryption system in Authorware.

External Content (External\External.a7p)

Provides an example of displaying external graphics based on a variable and displaying a graphic from a URL.

Flash in ActiveX (flash_ax\FlashAx.a7p)

Demonstrates messaging from a Flash 5 movie into an Authorware piece, using the ActiveX control.

Forms (Forms.a7p)

Shows how to use a framework; includes a list to create a multifield form.

Forms with WinCtrls (Forms_Win.a7p)

Demonstrates how to set up a multi-field form using the Windows Control Knowledge Objects.

Free Form Text (Q_txt_l.a7p)

Illustrates simple methods for evaluating text responses that may not have an easily defined answer.

Generic Question Model (Q_model.a7p)

Shows how to create a generic multiple-choice interaction that can be used as a model for rapid development.

Glossary (Glossary.a7p)

Demonstrates a simple glossary structure. Demonstrates the use of external data files, WordClicked, LineClicked, and string manipulation. The glossary functions include a scrolling list of terms and the ability to click on any word on the screen to search for its definition.

Graphing (Graph.a7p)

Demonstrates graphing simple data points using the Line function. The user can enter values for the data points either by clicking on the grid or keying in numbers for x and y .

Hangman (Hangman.a7p)

Shows you how to create a hangman game.

Help System (Help.a7p)

Demonstrates a simple help system using Frameworks. From the help screen, the user can select a topic by clicking on it in the list, or use the framework Find function to navigate to a help topic.

Hot Object Interaction (Q_hotobj.a7p)

Discusses clickable object interactions and how to set them up with a catchall.

Hot Spot Interaction (Q_hotspt.a7p)

Demonstrates the use of hot spots, how to layer hot spots, and catchall hot spots.

Hot Text (Hottext.a7p)

Demonstrates the use of hot text links.

Jigsaw Puzzle (Jigsaw.a7p)

Shows how to create a jigsaw puzzle in Authorware.

Judging and Tracking (Judge.a7p)

Shows how to judge and track a student's response at an interaction.

Keyboard (Keyboard.a7p)

Shows how to create an onscreen keyboard.

Keypress Interactions (Keypress.a7p)

Demonstrates setting up a keypress interaction and how to check for special keys. The example shows how to capture each keystroke and display a bullet character. The Logon piece uses this same interaction for the password entry.

Logon (Logon.a7p)

Shows how to prompt the user for a name and password. The example checks for existing records and resumes any previous file. If the logon doesn't find any records, the user can sign in, retype the name, or quit.

Macromedia Flash Asset Xtra (Flash.a7p)

Demonstrates how to use the Flash Asset Xtra in Authorware.

Magnify (Magnify.a7p)

Uses the Authorware graphic scaling ability and external content support to create a graphic zooming function.

Making Icons Immovable (Lockdown.a7p)

Shows how to lock down the contents of an icon.

Marching Ants (Marchant.a7p)

Demonstrates using the `Line` function to simulate a marching-ants marquee. It also demonstrates how to select an object with the marquee.

Memory Game (Memgame.a7p)

Demonstrates how to build a memory game.

Morse Code (Morse.a7p)

Demonstrates how to build a Morse-code generator. The user can convert text to Morse code or create tones by holding down the mouse button.

Movie Controls (Movctrls.a7p)

Illustrates how to create controls for any digital movie.

Moving Objects (Move.a7p)

Shows how to animate and position graphics. Includes a rotating gauge, a progress bar, and a scrolling graphic.

Mr. ASCII Pseudo (Ascii.a7p)

Demonstrates how to create an animation using character strings.

Navigating & Expressions (Nav_page.a7p)

Shows how to use an expression in a navigation icon to navigate to a page. The user can click on a list of icon titles to navigate to an icon.

Navigating with Flash (flash_nav\Nav_Flash.a7p)

Show how to use a Flash file as a navigation bar within an Authorware file.

Notepad (Notepad.a7p)

Illustrates a combination of functions such as reading/writing files and working with strings. Functionally, the notepad lets a user enter, view, and delete notes.

Orbit (Orbit.a7p)

Shows how to animate an object so that it goes behind and in front of the same object it travels around.

Paging (Paging.a7p)

Provides a simple paging structure setup. In addition to simple paging, the example illustrates preventing wraparound paging.

Picture Box (Pic-box.a7p)

Demonstrates a game in which a picture is divided into nine pieces. Eight of the pieces are displayed in random order on a 3 x 3 grid. The user can slide one piece at a time into the empty space. The objective is to rearrange the pieces into their proper order.

Pie Chart (PieChart.a7p)

Uses the `Line` function to draw many single lines. Together, the lines look like a colored pie chart.

Piston (Piston.a7p)

Demonstrates how to create motion similar to that of a piston.

Pop-up Glossary (Popup.a7p)

Shows how to create a pop-up window to display text.

Pull-down Menus (Pulldown.a7p)

Shows how to create menu separators, create accelerator keys, and remove the File menu.

Radio Buttons (Radiob.a7p)

Shows how to set up radio button logic.

Rat Maze (Ratmaze.a7p)

Demonstrates how to build a maze game.

RealNetworks (Real.a7p)

Shows how to use the RealPlayer Control for ActiveX in Authorware.

Referring to Icons (Refer.a7p)

Shows how to refer to the value of a variable at an icon.

RGB (RGB.a7p)

Shows how to use RGB to change the color for the drawing functions.

Right and Left Clicks (Rightclk.a7p)

Demonstrates how to create a hot spot that responds to right and left mouse clicks.

Rotating Graphic (Rotate.a7p)

Demonstrates how to rotate a graphic based on the position of a slider or a value that the user enters.

RTF Objects Editor (RTF\RTF.a7p)

Shows how to use the RTF Objects Editor to create a media rich RTF file within Authorware with graphics, variables, expressions, and hypertext links.

Screen Capture (screencp\Screencp.a7p)

Shows how to capture screens from within Authorware.

Screen Saver (Ssaver.a7p)

Demonstrates how to create a screen saver in Authorware.

Screening Text or Numbers (Screen.a7p)

Provides methods for extracting the characters desired from a string. In the example, the user keys in text, and it is split into numeric and nonnumeric strings.

Scripting & Expressions (Script.a7p)

Provides basic examples of if...then statements, repeat loops, and the `Test` function.

Scrollable Lists (Scrllist.a7p)

Demonstrates how to create a scrollable, highlighting list box.

Scrolling Checklist (Scrollick.a7p)

Demonstrates a scrolling checklist: lines of scrolling text with each line preceded by a field that you can click to cycle through two or more values.

Set Motion Object (setmotionobject.a7p)

Demonstrates how you can use one motion icon to move a number of separate objects.

Short Text Interaction (Q_txt_s.a7p)

Shows how to create short text interactions so that they anticipate all possible responses.

Show Me Credits (Credits.a7p)

Shows how to set up names of the people who contributed to the development of the Show Me files.

Slider (Slider.a7p)

Shows how to set up a simple slider.

Smart Glossary (Smart.a7p)

Shows how to set up a general hypertext style that jumps from any term to the appropriate definition, using the Authorware `FindText` and `PageFoundID` functions.

Snake (Snake.a7p)

Demonstrates how to create a snake game.

Storing/Retrieving Information (Write.a7p)

Shows how to read, write, and append text to a file. The user enters text in a text response and then can select a button to write, append, or read the text from the text file.

Tab Buttons (Tabs.a7p)

Demonstrates how to create tabs, a popular metaphor for organizing information.

Target Area Interaction (Q_target.a7p)

Shows how to set up drag and drop for multiple objects, and the use of a catchall.

Ticker Tape (Ticker.a7p)

Demonstrates how to have a line of text move in a horizontal cycle, like a ticker tape.

Time Limit Interaction (Q_time.a7p)

Demonstrates setting up a time limit at an interaction.

Tool Palette (Toolpal.a7p)

Shows how to create a movable tool palette.

Tool Tips (Tooltips.a7p)

Shows how to create tooltips—those little yellow reminders that pop up when you move the pointer over a button.

Tries Limit Interaction (Q_tries.a7p)

This example uses a musical quiz to demonstrate how to limit the number of chances for a user to select an option.

Using ActiveX (activex\Activex.a7p)

Shows how to insert and use ActiveX controls in Authorware. Includes demonstrations of a web browser, a calendar, and the Macromedia Flash control for ActiveX.

Using Lists (Lists.a7p)

Shows how to create and use lists in Authorware. Also shows how to sort lists.

Using ODBC (Odbc\Odbc.a7p)

Shows how to connect to a data source through ODBC. This piece includes sample SQL statements to store, retrieve, open, and close a data source.

Using Scripting Xtras (Xtraio.a7p)

Shows how to create a child instance of the File IO Xtra for reading and writing a file.

Variable Branching (Decision.a7p)

Shows how to determine a decision path with a variable or expression.

Variable Button Labels (vaributt.a7p)

Demonstrates how to place variable text over a button.

Window Resize (Winsize.a7p)

Demonstrates how to create a presentation window that the user can resize at runtime.

XML Parser (XMLParser.a7p)

Shows how to use the XML Parser scripting Xtra (which must be installed in your Xtras folder.)

CHAPTER 13

Using Windows Controls Knowledge Objects and WinCtrls.u32

The WinCtrls.u32 file lets you display common Windows controls in your Macromedia Authorware piece. Windows controls are standard interface elements (widgets) with which the user can interact. Using a control is a matter of selecting a control type and then setting or retrieving properties for that control.

You can include controls in an Authorware piece either by using the supplied Knowledge Objects or by hand-coding.

About WinCtrls.u32

The Windows operating system already has the capability to display standard controls such as buttons, list boxes, and check boxes. Before the WinCtrls.u32 file, you had to use a tedious process to simulate these controls in Authorware. Now, the WinCtrls.u32 file provides a simple interface to these controls.

When you distribute an Authorware piece that includes a Windows control, you must include the WinCtrls.u32 file with the Authorware piece.

Here are some of the features of WinCtrls.u32:

- Support for multiple instances
- Internal tab order and tab handling
- Support for the Authorware erase icon
- Support for the Authorware motion icon
- Open, flexible architecture (new controls can be added in the future)
- Keypress interactions that remain active when a control has the focus (controls don't "trap" any key presses)
- Easy event handling (no need to initialize the event variable after an event has occurred)
- Optional use of the Authorware internal Paste Hand
- Control over colors, fonts, visibility, and appearance
- Support for standard Windows tooltips
- Authorware Web Player support
- Clipboard support
- Right mouse button support
- Small file size and low memory overhead

Windows controls offer some advantages over ActiveX controls, including the following:

- Unlike an ActiveX control, if you put a Windows control in a web piece, you can run it in Netscape Navigator.
- ActiveX controls must be registered, but you do not need to register Windows controls.
- You cannot redistribute Microsoft Forms 2.0 ActiveX controls unless you are a registered Microsoft developer, but anyone can distribute Windows controls.
- You can redistribute one file (WinCtrls.u32) instead of multiple OCXs.
- You don't have to worry about version conflicts with Windows controls.
- The WinCtrls.u32 file returns data types that Authorware can handle.
- WinCtrls.u32 supports internal tab order automatically.

Before you begin

Before you use the WinCtrls.u32 file, you should know how to use the Authorware flowline and icons. Make sure you understand the difference between an icon's title and its ID. You should also know how to create basic scripts in Authorware, load custom functions, and create user variables.

When you use the WinCtrls.u32 file, you can create a carriage return by using `\r` within a text string. This is equivalent to the Authorware variable `Return`. For example,

```
"Line 1\rLine 2\rLine 3"
```

and

```
"Line 1" ^ Return ^ "Line 2" ^ Return ^ "Line 3"
```

are equivalent, and both result in the following display:

```
Line 1  
Line 2  
Line 3
```

In addition, be aware of the following limitations:

- You cannot have a transparent control.
- You cannot layer a control in Authorware. A control is always drawn direct to screen.

About Windows Control Knowledge Objects

You use Windows Control Knowledge Objects to display controls in your Authorware piece. The WinCtrls.u32 file provides three Knowledge Objects:

Windows Control Knowledge Object This Knowledge Object creates a single instance of a control. (If you want multiple instances, you must use multiple Knowledge Objects.) You use this Knowledge Object to create a control before you attempt to get or set properties.

For more information, see [“Using a Windows Control Knowledge Object to create a Windows control” on page 233](#).

Set Property Knowledge Object After you have created a control, you use this Knowledge Object to set the various attributes (called *properties*) for that control.

For more information, see [“Setting a control's properties with the Set Property Knowledge Object” on page 236](#).

Get Property Knowledge Object You use this Knowledge Object to retrieve the current value of a control's property. The value is placed in a user variable that you create or select. Each Knowledge Object can retrieve only one value.

For more information, see [“Retrieving a control’s properties with the Get Property Knowledge Object” on page 237.](#)

Using a Windows Control Knowledge Object to create a Windows control

You use the Windows Control Knowledge Object to create a control. After creating the control, you use the Set Property Knowledge Object to set the value of the control's properties.

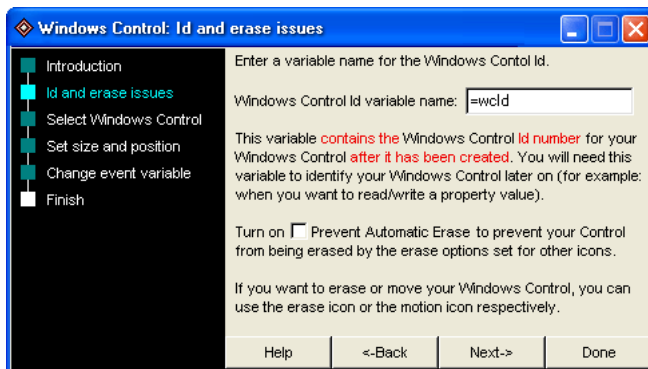
To use a Windows Control Knowledge Object to create a Windows control:

- 1 Drag a Windows Control Knowledge Object from the Knowledge Objects window onto the flowline at the point where you want to display a control.

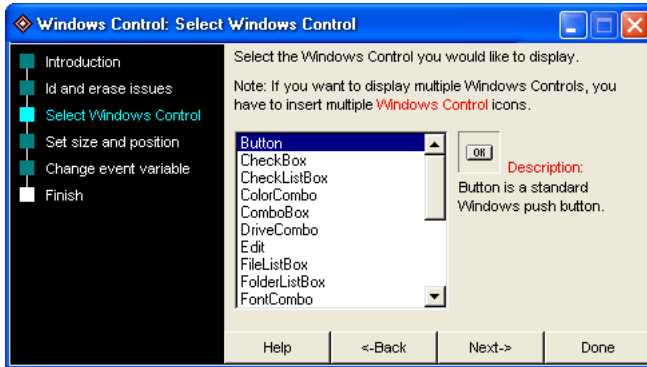
The Windows Control: Introduction screen appears.

- 2 Click Next, or select Id and Erase Issues, to display the Windows Control: ID and Erase Issues screen, then enter the name of the user variable that will be used when displaying the control. (ID variables are used by Authorware to identify controls.)

Authorware automatically erases content in many situations. For example, exiting the branch of a decision icon that has been set to Erase Contents: Upon Exit will erase any content displayed in the branch. Windows controls will also be automatically erased. To prevent a control from automatically erasing, select Prevent Automatic Erase.

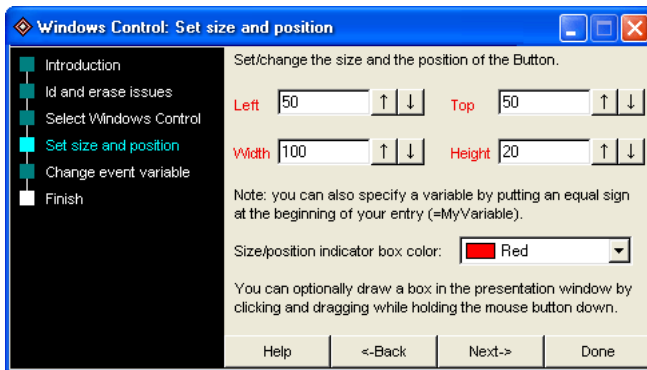


- 3 Click Next, or select Select Windows Control, to display the Windows Control: Select Windows Control screen, then select the type of control you want to display.



For information about each control, see Types of controls in Authorware Help.

- 4 Click Next, or select Set Size and Position, to display the Windows Control: Set Size and Position screen.



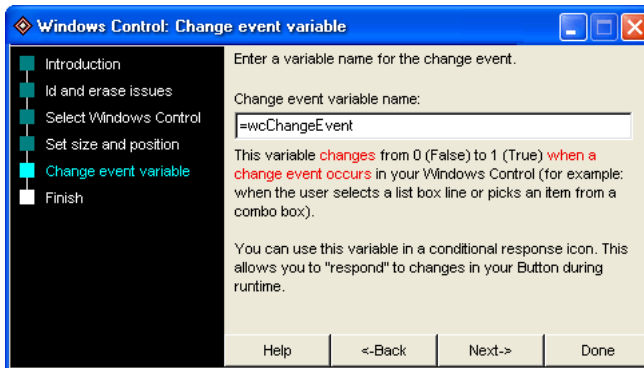
This screen, which appears in front of the Presentation window for your Authorware piece, lets you set the size and the position of the control. (The height of some controls—for example, ComboBox—is fixed in proportion to the font size used.) In the Presentation window, you can see a box whose color matches the color in the Size/Position Indicator Box Color list. This box indicates the current size and position of the control.

- 5 If you want to change the size and position settings, drag within the Presentation window to redraw the box.

The changes are reflected in the settings of the Windows Control: Set Size and Position screen.

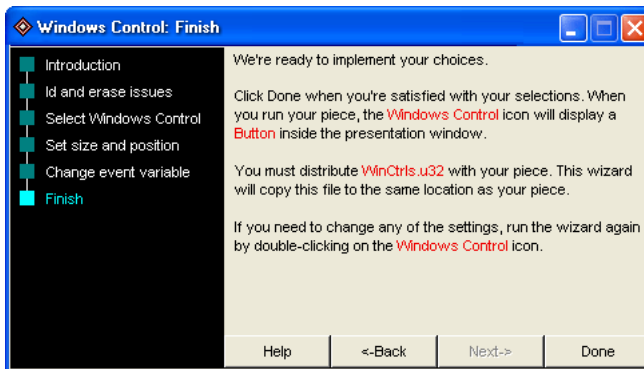
- 6 Click Next, or select Change event variable, to display the Windows Control: Change Event Variable screen, then enter the name of a user variable that will be triggered when a change event occurs.

A change event occurs when the status of the control changes; see [“Using the change event variable name” on page 246](#).



The use of a change event variable is optional.

- 7 Click Next, or select Finish, to display the Windows Control: Finish screen, which displays the name of the control that you specified. Click Done to apply your settings, or click Back to edit your settings.



Setting a control's properties with the Set Property Knowledge Object

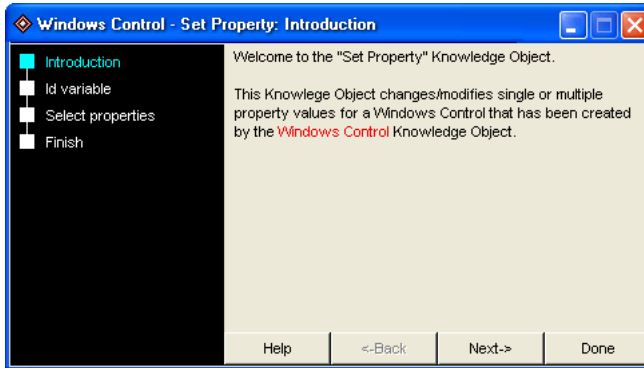
After you create a control to display, you can modify (or set) its properties. If you do not set a control's property, the property's default value is used.

Some properties depend on the values of other properties. For example, setting `Enabled` to `true` will appear to have no effect if `Visible` is set to `false`.

To set property values:

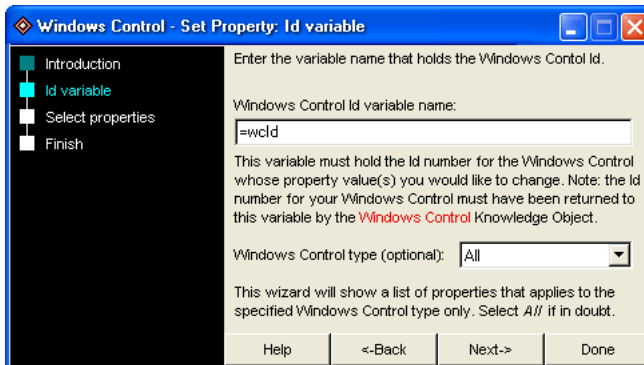
- 1 Drag a Windows Control—Set Property Knowledge Object from the Knowledge Objects window onto the flowline at the point where you want to set property values.

The Windows Control—Set Property: Introduction screen appears.



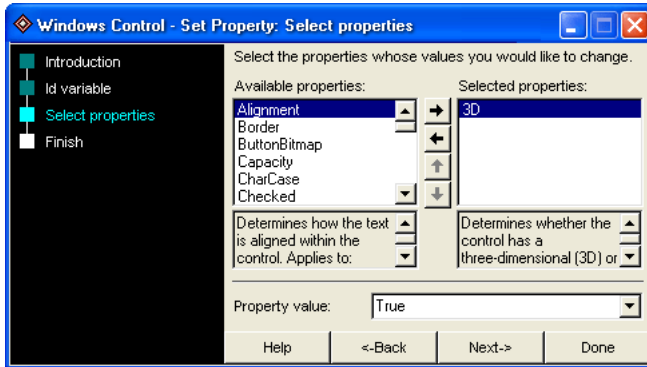
- 2 Click Next, or select Id variable, to display the Windows Control—Set Property: ID Variable screen, and enter the user variable that you used to display the control. (ID variables are used by Authorware to identify controls.)

If you know the type of control, you can select it from the Windows Control Type list. This restricts the properties that appear in the following screen to those that are relevant to the selected control type.

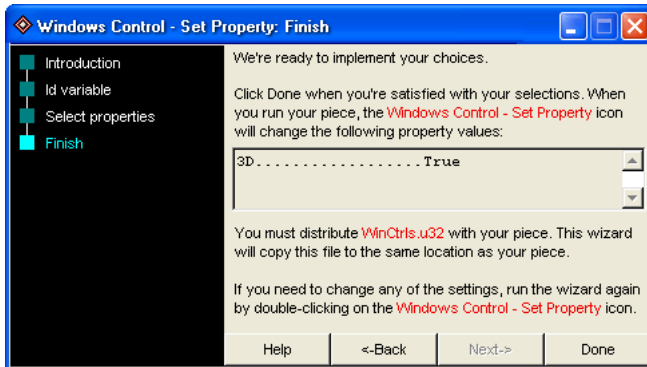


- 3 Click Next, or select Select properties, to display the Windows Control—Set Property: Select Properties screen, and select a property in the Available Properties list. Then click the right arrow to move it to the Selected Properties list.

- 4 In the Selected Properties list, click the property once to highlight it. In the Property value text box, select or enter a value.



- 5 Click Next, or select Finish, to display the Windows Control—Set Property: Finish screen, which shows a summary of the property values that will be applied to the control. Click Done to apply these values, or click Back to edit the values.



Retrieving a control's properties with the Get Property Knowledge Object

Your Authorware piece may need to retrieve the current value of a Windows control property. For example, you might need to retrieve the item that is currently selected in a list. The Get Property Knowledge Object places the retrieved property value into a user variable.

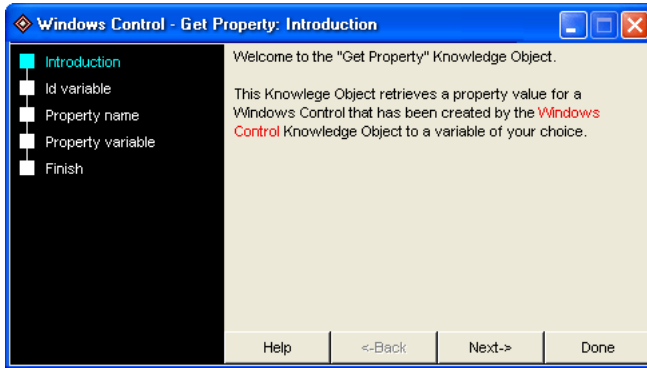
You can retrieve the value of a control's property only while that control currently exists. This means that although the control doesn't necessarily need to be visible, it must not have been erased before you attempt to retrieve the property value.

After you decide which property value you want to retrieve, you must determine where in the Authorware flowline you want it retrieved. For example, you might want to retrieve the value within an interaction icon's response if that response was triggered by a user event such as a mouse click.

To retrieve a property value:

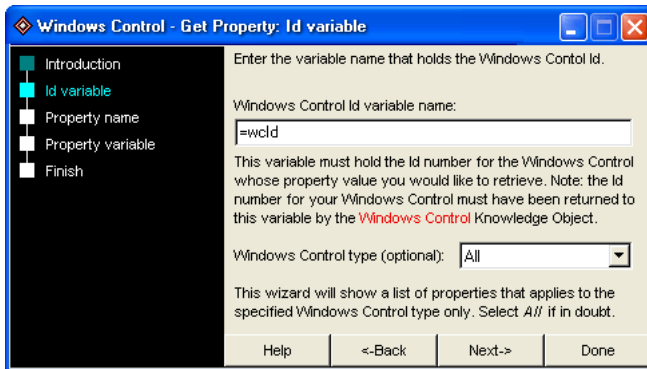
- 1 Drag a Windows Control—Get Property Knowledge Object from the Knowledge Objects window onto the flowline at the point where you want to retrieve the value.

The Windows Control—Get Property: Introduction screen appears.



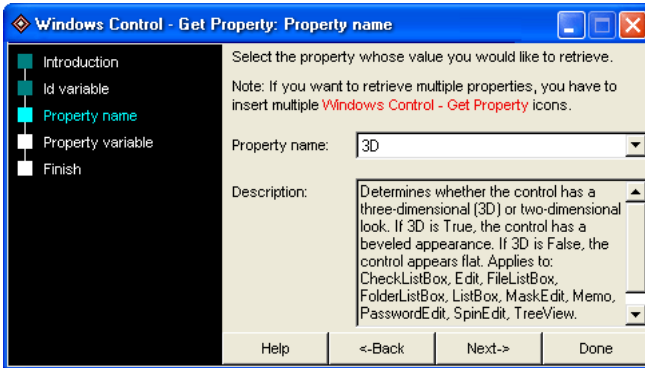
- 2 Click Next, or select Id variable, to display the Windows Control—Get Property: ID Variable screen, and enter the name of the user variable you used to display the control. (ID variables are used by Authorware to identify controls.)

If you know the type of control, you can select it from the Windows Control Type list. This restricts the properties that appear in the following screen to those that are relevant to the selected control type.



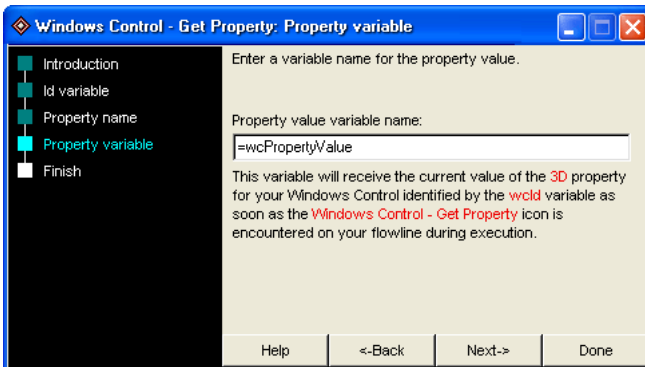
- 3 Click Next, or select Property name, to display the Windows Control—Get Property: Property Name screen, and select the property whose value you want to retrieve.

You can retrieve only one property's value. To retrieve more than one value, you must use a separate Knowledge Object for each value.

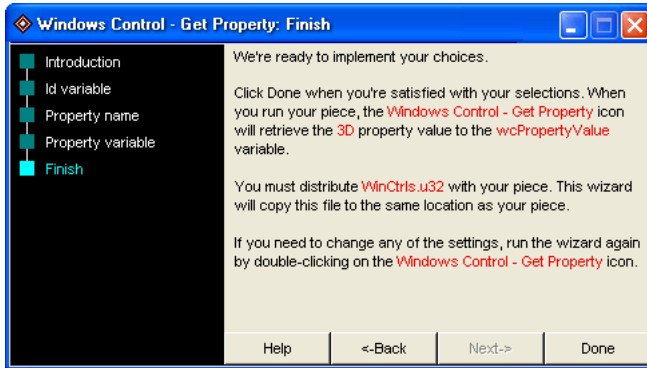


- 4 Click Next, or select Property variable, to display the Windows Control—Get Property: Property Variable screen, and enter the name of an Authorware user variable.

In your Authorware piece, when this Knowledge Object is encountered, this user variable will be updated to reflect the retrieved value.



- 5 Click Next, or select Finish, to display the Windows Control—Get Property: Finish screen, which shows a summary of the changes that will be made to the Knowledge Object. Click Done to apply these changes, or click Back to edit the changes.



Example: Building a list box

The steps for building Windows controls using Knowledge Objects follow the same basic procedure: You create a control, set its properties, and check the state of the properties.

The following basic steps describe how to use Knowledge Objects to build a Windows list box.

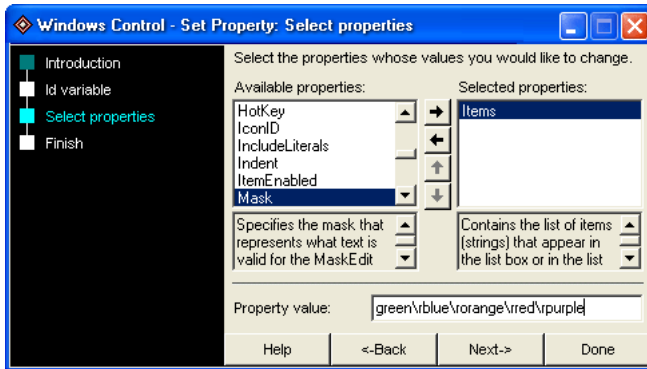
To display the ListBox control:

- 1 Drag a Windows Control Knowledge Object to the flowline.
The Windows Control: Introduction screen appears.
Note: You will be prompted to save your file if you have not previously saved it.
- 2 Click Select Windows Control.
The Windows Control: Select Windows Control screen appears.
- 3 Select ListBox for the type of control.
- 4 Click Next to display the Windows Control: Set Size and Position screen. Set the dimensions for the list box, and leave all other settings at their defaults.
- 5 Click Done.
The following message appears: “The wizard is now configuring the Knowledge Object. Please wait.” When wizard is finished, the Knowledge Object appears on the flowline.
- 6 On the flowline, name the Knowledge Object **My ListBox**.

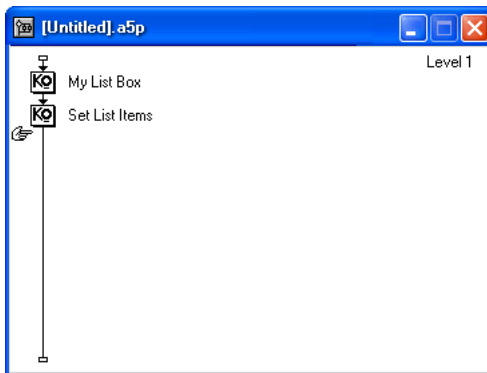
To set the control's properties:

- 1 Drag a Windows Control—Set Property Knowledge Object to the flowline, after My ListBox.
The Windows Control—Set Property: Introduction screen appears.
- 2 Click Select Properties to display the Windows Control—Set Property: Select Properties screen. In the Available Properties section, select the **Items** property, and click the right arrow to move it to the Selected Properties section.

- 3 In the Property Value text box, type `green\rblue\rorange\rred\rpurple`.



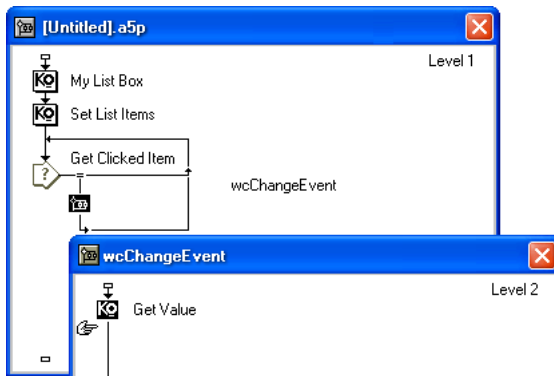
- 4 Click Done.
- 5 On the flowline, name the Knowledge Object **Set List Items**.



To retrieve the ListBox control's property value:

- 1 Drag an interaction icon to the flowline, and name it Get Clicked Item.
 - 2 Attach a map icon as a conditional response, and name it `wcChangeEvent` (the default name of the wizard variable).
 - 3 In the Conditional tab of the Properties window, set the conditional response to Automatic: On false to true.
 - 4 Open the Map and drag a Windows Control—Get Property Knowledge Object to the flowline. The Windows Control—Get Property: Introduction screen appears.
 - 5 Click Property Name to display the Windows Control—Get Property: Name screen. From the pop-up menu, select Value.
 - 6 Click Next to display the Windows Control—Get Property: Property Variable screen. Make sure the value of the variable is `=wcPropertyValue`.
 - 7 Click Done. If a message appears alerting you that the `wcPropertyValue` variable will be created, click Yes.
- When the Knowledge Object is created, it appears on the flowline.

- 8 On the flowline, name the Knowledge Object **Get Value**.



- 9 Drag a calculation icon to the flowline and name it **myitems**. Open it, and type something like the following to define a new user variable that returns a delimited list called **myitems**:

```
myitems:="item1\ritem2\ritem3\ritem4"
```

- 10 Directly under **Get Value**, drag a calculation icon to the flowline. Open it, and type the following:

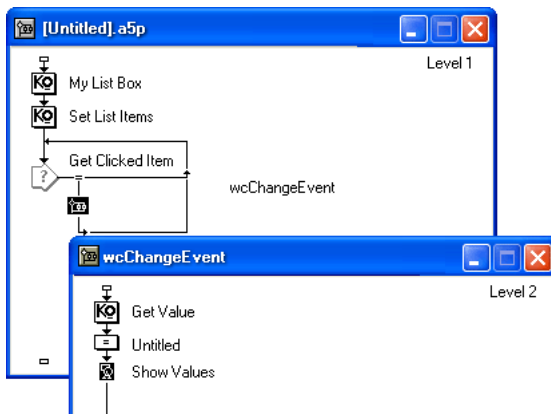
```
clicked := GetLine(myItems, (wcPropertyValue + 1) )
```

Note: The list item numbering is zero-based—that is, the first item is 0, the second is 1, the next item is 2, and so on.

- 11 Drag a display icon to the flowline, name it **Show Values**, and open it. Use the Text tool to type the following:

```
value: {wcPropertyValue} list item: {clicked}
```

When you are finished, your flowline should look like this:



- 12 Save your Authorware piece.

You're done! You've used Windows controls to display a list box and to retrieve the text of the selected ListBox item.

Available Windows controls and properties

There are several different Windows controls, each of which has valid properties that can be read or set. The following table displays all the currently available controls and properties. An *X* indicates that the property is valid for that control type.

For more information, see Types of controls and Windows control properties in Authorware Help.

Property	Button	CheckBox	CheckListBox	ColorCombo	ComboBox	DriveCombo	Edit	FileListBox	FolderListBox	FontCombo	ListBox	MaskEdit	Memo	Menu	PasswordEdit	RadioButton	SpinButton	SpinEdit	TabSet	TrackBar	TreeView
3D			X				X	X	X		X	X	X		X			X			X
Alignment		X											X	X		X		X			
Border			X				X	X	X		X	X	X		X			X			X
ButtonBitmap	X																				
Capacity													X								
CharCase				X	X	X	X			X		X	X		X			X			
Checked			X											X							
Clear			X	X	X	X	X	X	X	X	X	X	X	X	X			X	X		X
Color		X	X	X	X	X	X	X	X	X	X	X	X		X	X		X	X	X	X
Columns			X					X	X		X										
ComboBoxStyle					X																
Cursor	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
DispatchKeys	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Drive						X		X	X												
DropDownCount				X	X	X				X											
Enabled	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Expanded																					X
Filter								X													
Focus	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
FocusEvents	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Folder								X	X												
FontColor	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X		X	X		X
FontName	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X		X	X		X
FontSize	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X		X	X		X
FontStyle	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X		X	X		X

Property	Button	CheckBox	CheckListBox	ColorCombo	ComboBox	DriveCombo	Edit	FileListBox	FolderListBox	FontCombo	ListBox	MaskEdit	Memo	Menu	PasswordEdit	RadioButton	SpinButton	SpinEdit	TabSet	TrackBar	TreeView
Handle	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
HideSelection							X					X	X		X		X				X
Hint	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
HotKey	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
IconID	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ImageIndex											X			X							X
Images											X			X							X
IncludeLiterals												X									
Indent											X										X
ItemEnabled			X											X							
ItemHeight											X										
Items			X	X	X	X		X	X	X	X			X					X		X
Mask												X									
MaskChar												X									
Max																	X	X		X	
MaxLength					X		X					X	X		X			X			
Min																	X	X		X	
MouseButton														X							
MultiSelect			X					X	X		X										
Orientation																				X	
PasswordChar															X						
RadioGroup																X					
ReadOnly							X					X	X		X			X			
RGB				X																	
ScrollBars											X		X								
SelCount			X					X	X		X										
Selected			X					X	X		X										
SelLength							X					X	X		X			X			
SelStart							X					X	X		X			X			
SelText							X					X	X		X			X			

Property	Button	CheckBox	CheckListBox	ColorCombo	ComboBox	DriveCombo	Edit	FileListBox	FolderListBox	FontCombo	ListBox	MaskEdit	Memo	Menu	PasswordEdit	RadioButton	SpinButton	SpinEdit	TabSet	TrackBar	TreeView
ShowHint	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ShowLines																					X
SliderLength																				X	
Sorted			X		X						X										
TabColorSelected																			X		
TabColor Unselected																			X		
TabStop	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Text	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X		X
TopIndex			X					X	X		X										
Value		X	X	X	X	X		X	X	X	X			X		X	X	X	X	X	X
VariableAware				X	X	X	X			X		X	X		X			X			
VariableName				X	X	X	X			X		X	X		X			X			
Visible	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
WordWrap													X								

For information about the types of controls, variables, and functions available, see Using Windows controls Knowledge Objects and WinCtrls.u32 overview in Authorware Help.

Using Authorware Windows controls variables

In general, the only types of Authorware user variables that you can use with Windows Control Knowledge Objects are regular and list variables. User icon variables are not currently supported, although you can use them if your controls are hand-coded. For more information, see [“Using user icon variables” on page 248](#).

To use a variable, precede the name with an equal sign (=), except in the Property Value field for VariableName. In this case, you do not precede the name with an equal sign if the name is the actual name of the variable to use.

Note: The normal rules for valid user variable names still apply. For example, variable names must begin with a letter.

Using variables in a Windows Control Knowledge Object

In a Windows Control Knowledge Object, a user variable is needed for the control ID. Similarly, user variables can be used to position a control and they are used to indicate Change Events.

These variable names are always preceded with an equal sign (=). You can use an existing variable, or you can create a new variable. If the name you enter for the variable refers to a user variable that does not exist, it is created.

Using variables in a Set Property Knowledge Object

In the Windows Control Set Property Knowledge Object, variables are used to set properties for a control. User variables can also be used for any property (except read-only properties). You enter the name of the variable in the Property Value field.

If you are setting the `VariableName` property and are entering the actual name of the user variable, do not precede the variable name with an equal sign. The variable will not be created—you must create the variable in your Authorware piece. You can still use a user variable, whose value is the name of the actual user variable, by preceding it with an equal sign in the normal manner.

Using variables in a Get Property Knowledge Object

You must include user variables to get properties from a control. Add the user variable name (this variable is the one that was originally used when the control was created) immediately after the equal sign (=) without any leading spaces. If the variable does not already exist, it is created. On some screens, the equal sign (=) is preset and cannot be removed. Add the variable name immediately after the equal sign (=) without any leading spaces. If the variable does not already exist, it is created.

Using the change event variable name

The variable `ChangeEventVarName` (or change event variable name) is a type of event—for example, a single click on an item in a control. The change event variable changes from 0 (`false`) to 1 (`true`) and then immediately back to 0. You can use this user variable in a conditional response (set to `On false to true`) so that Authorware can respond to the event of the user clicking a button.

When the user double-clicks a control (for example, an item in a list box), the change event variable changes from 0 to 2. Although a value of 2 is still `true` (any positive number is `true`), you can distinguish between a single click and a double click.

The use of a change event variable name is optional. Use it when necessary; it has a small effect on performance.

You should not explicitly set the change event variable to 0 immediately after it has been triggered because it might make `WinCtrls.u32` reset the variable to 1. In general, you should not have to explicitly set the change event variable's value, because `WinCtrls.u32` controls its value.

Using List Variables

If you are using list variables with Windows controls, the list variables must be given the appropriate initial value; otherwise Windows controls might not update the variable's value correctly. You can select the variable name in the Variables dialog box and edit the Initial value field. Linear list variables should be initialized as `[]` and property list variables should be initialized as `[:]`.

About the control ID number

Each time a control is created, it can be given a unique ID number that is usually assigned to an Authorware user variable. If you use the `wcDisplayControl` function in a calculation icon, you aren't required to assign the ID to a variable; however, if it isn't assigned, you cannot reference the control—for example, to set a property.

The number is actually the return value from the function that creates the control. Authorware can communicate with a control: it does so by referencing the control's ID number (by means of the variable).

The following example is a simplified representation of a typical communication between Authorware and the WinCtrls.u32 file:

Authorware: WinCtrls, display an edit control that I'll call *foo*.

WinCtrls: OK, I've done it and I've called it 1234. (That is, *foo* is 1234.)

Authorware: WinCtrls, tell "*foo*" to change its font to Arial.

or

```
foo := wcDisplayControl( 400, 120, 300, 20, "Edit" )
wcSetPropertyValue( foo, "FontName", "Arial" )
```

You need to create the control before you can set any of its properties. Also, you must not set two current controls to the same Authorware user variable.

Coding a control

Before you start, check that the WinCtrls.u32 functions `wcDisplayControl()`, `wcGetPropertyValue()`, and `wcSetPropertyValue()` have been loaded into your Authorware piece.

Creating a control

In general, use code like this to create a control within a calculation icon:

```
control Var := wcDisplayControl( x1, y1, width, height, control
name[, change event variable name] )
```

This code uses the following arguments:

- *control Var* is the Authorware user variable that will contain the ID number of the control after it has been created.
- *x1* is the *x* coordinate of the upper left corner.
- *y1* is the *y* coordinate of the upper left corner.
- *width* is the pixel width.
- *height* is the pixel height.
- *control name* is the type of control.
- *change event variable name* is name of the Authorware user variable that is to change from 0 (false) to 1 (true) each time a change event occurs.

The following example creates a control:

```
ctrl_ID := wcDisplayControl( 125, 60, 325, 150, "Memo",
"ctrl01Change" )
```

Displaying a control

To display a control, use code like this within a calculation icon:

```
control Var := wcDisplayControl( x1, y1, width, height, control name, [ change  
event variable name] )
```

The following example displays a control:

```
ctrl_ID := wcDisplayControl( 125, 60, 325, 150, "Memo",  
"ctrl01Change")
```

Setting the property of a control

To set or change a control's property, make sure you have already created the control and used an Authorware variable to capture the resulting return value. This return value is the control's ID number. Typically, the code to change a property looks like this:

```
Result := wcSetPropertyValue( ID number, property name, Value )
```

For more information, see the function `wcSetPropertyValue` in Authorware Help.

The following example changes a control's property:

```
ctrl_ID := wcSetPropertyValue( ctrl_ID, "MaxLength", "100")
```

The `WinCtrls.u32` file should typecast property values for you. For example, each of the following lines shows two equivalent values:

```
1, "1"  
"True", TRUE
```

Similarly, each of the following lines shows two equivalent values:

```
0, "0"  
"False", FALSE  
100, "100"
```

Reading a property's current value

To get a property's value, the control must currently exist. Typically, your query would look like this:

```
Value := wcGetPropertyValue( ID number, property name )
```

The following example reads a property's value:

```
myFont := wcGetPropertyValue( ctrl_ID, "FontName" )
```

For more information, see the function `wcGetPropertyValue` in Authorware Help.

Using user icon variables

If you are an experienced authors who is familiar with scripting in Authorware, you can hand-code a control. When hand-coding a control, you can use user icon variables for both the control ID and the change event variable name.

For example, if there is an icon titled "My Settings", a variable `bar@"My Settings"` could be used as the control ID like this:

```
bar@"My Settings" := wcDisplayControl(100, 100, 200, 120, "Memo")  
wcSetPropertyValue(bar@"My Settings", "FontName", "Arial")
```


If the user icon variable is also a list variable, it could be used like this:

```
bar@"My Settings"[1] := wcDisplayControl(100, 100, 200, 120, "Memo")
wcSetPropertyValue(bar@"My Settings"[1], "FontName", "Arial")
```

For the variable `foo@"My Settings"` to be used as the change event variable name, the variable needs to be in a form that is evaluated correctly. That is, the double quotation marks (") need to be preserved.

For example, the following variables are all invalid:

```
newCtrl := wcDisplayControl(100, 100, 200, 120, "Memo", "foo@"My
    Settings")
newCtrl := wcDisplayControl(100, 100, 200, 120, "Memo", "foo@"My Settings")
newCtrl := wcDisplayControl(100, 100, 200, 120, "Memo", "foo@My Settings")
newCtrl := wcDisplayControl(100, 100, 200, 120, "Memo", "foo@"My
    Settings\")
```

Authorware will incorrectly see `My Settings` as a user variable, return syntax errors, or simply not work.

You must “escape” the double quotation marks (") so that Authorware recognizes them as part of the variable name and not as the ends of text strings; you must also add double quotation marks (") to redefine the text strings. One way is to use `Char(34)`, the Authorware equivalent of the double quotation mark ("), in place of the user variable’s quotation marks:

```
"foo@" ^ Char(34) ^ "My Settings" ^ Char(34)
```

If the variable is also a list variable, you could use code like this:

```
"foo@" ^ Char(34) ^ "My Settings" ^ Char(34) ^ "[1]"
```

A simpler way is to precede the double quotation marks (") that need to be preserved with a backslash (\):

```
"foo@"@"My Settings\""
```

If the variable is also a list variable, you could use code like this:

```
"foo@"@"My Settings\"[1]"
```

For example, the following two code samples show the two methods of using the name of the user icon variable `foo@"My Settings"` for the change event variable:

```
newCtrl := wcDisplayControl(100, 100, 200, 120, "Memo", "foo@"^Char(34)^"My
    Settings"^Char(34))
newCtrl := wcDisplayControl(100, 100, 200, 120, "Memo", "foo@"@"My Settings\"
)
```

Note: The normal rules for valid user variable names still apply. For example, variable names must begin with a letter.

CHAPTER 14

Using JavaScript in Authorware

In Macromedia Authorware 7, authors have a new choice of programming languages to use within calculation icons. You can continue to use the original language (Authorware Script Language, or AWS), or you can use support for JavaScript (JS), version 1.5, which is new in Authorware 7.

To successfully use JavaScript in Authorware, you need to know the following:

- **JavaScript programming** This chapter does not include information about how to program in JavaScript.
- **JavaScript version** You should be familiar with the version of JavaScript implemented in Authorware 7, its core functionality, and the language extensions used.
- **Authorware extensions** There is new functionality in Authorware that works with JavaScript.
- **How To's and examples** A number of JavaScript and Authorware Script tips and examples are provided.

About JavaScript

The dialect of the JavaScript programming language implemented in Authorware 7 is based on the ECMAScript Language Specification, Standard ECMA-262. This is the third edition of the specification; it corresponds to the version of the language understood by the Authorware JavaScript interpreter, version 1.5. At the time this manual was written, the core specification was available in a free download from the ECMA website at www.ecma-international.org/publications/files/ecma-st/ECma-262.pdf.

The goal in this release was to introduce to Authorware a conforming implementation of ECMAScript as defined in the ECMA-262 Standard, plus the necessary additional functionality needed to fully integrate with Authorware.

Note: The ECMA-262 specification *does not* include browser-specific extensions, such as `Document.Write()`. For more information, see [“Defining Authorware” on page 253](#).

The entire core JavaScript language is implemented in Authorware. It includes the following:

- Data types (the full range of values and precision)
- Expressions
- Operators

- Statements
- Function definitions
- Native ECMA script objects
- The global object
- Object objects
- Function objects
- Array objects
- String objects
- Boolean objects
- Number objects
- The math object
- Date objects
- RegExp (regular expression) objects

Authorware Document Object Model

In Authorware, the Document Object Model (DOM) is used to gain access to and manipulate icons and variables within the course.

The Authorware DOM defines the structure of a course that is created using icons. By representing icons and variables as objects and properties, the Authorware DOM provides a way for the course and its icons to be accessed and manipulated by the JavaScript language.

The structure of an Authorware course can be seen as a tree. There's a root icon, map icons, and offshoots of map icons that include display icons, motion icons, and other icon types.

In a DOM, the tree structure is preserved and presented as a hierarchy of parent icons and child icons. At each level within the course structure, the icon can be exposed to JavaScript as an object. Using this structure, you can access any icon in the course.

In JavaScript, you can gain access to any icon using the following JavaScript syntax:

```
var myIcon = new Icon(aw.IconID("MyIconTitle"))
```

It is important to distinguish between the DOM of an HTML document and the Authorware DOM. JavaScript support in Authorware doesn't have access to the objects in the DOM of an HTML document (and vice versa), since an Authorware piece isn't an HTML document.

The DOM of an HTML document includes objects like Anchor, Applet, Area, Button, Checkbox, and so on. These are meaningful in the context of an HTML document only. The Authorware DOM includes objects like aw and Icon. These are meaningful in the context of an Authorware piece only. Both DOMs include objects like Array, Boolean, Date, Function, and Math, because these objects are built into the JavaScript language itself (not part of any DOM).

If you are familiar with JavaScript in browsers, you can reference icons in your course in almost the same way that you reference objects in an HTML document. To reference icons in the Authorware course, however, you must call `aw.IconID("IconTitle")`, `new Icon(IconID)`, or another function that returns an icon ID or object. Then you can store the icon ID or object in a variable and use that variable in future references.

Authorware JavaScript language extensions

In addition to the wealth of functionality native to JavaScript, there are a number of Authorware-specific language extensions:

The *aw* object provides access to all of the Authorware functions and variables, including icon variables, and user functions and variables.

The *icon* object provides access to all of the properties and methods of any icon in the current file.

The Calculation editor is enhanced with an Authorware Script / JavaScript language selector, and JavaScript language syntax highlighting.

For details, see [“The *aw* object” on page 255](#).

Defining Authorware

Scripting languages, such as JavaScript or Authorware Script Language, are implemented or hosted in a larger application such as Authorware, a web browser or server, or a productivity application. Each host application extends the core language functionality appropriately for that application. Authorware includes the *aw* and *icon* objects, functions, variables, user-defined variables, Xtra extensions, U32s, and so on.

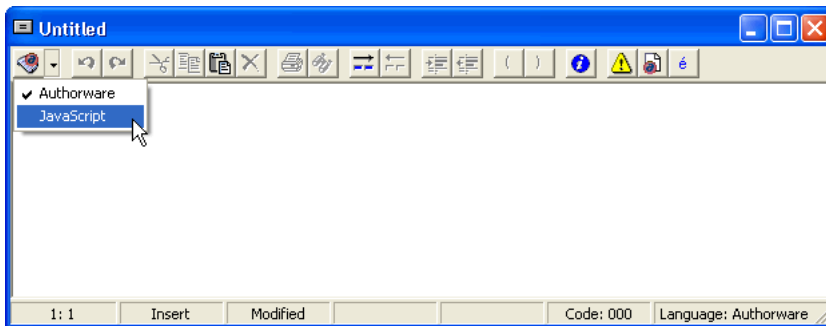
Experienced JavaScript developers usually work with the language within a web browser or server. Since Authorware is not a web browser, browser-specific elements, properties, methods, and objects are not available.

Entering JavaScript code

Entering JavaScript code in an Authorware calculation icon is similar to entering Authorware Script code.

To enter JavaScript code in an Authorware calculation icon:

- 1 Add a calculation icon to the flowline of a file.
- 2 Open the calculation icon.
- 3 In the upper left corner, click the language icon to display the choices, and select JavaScript.

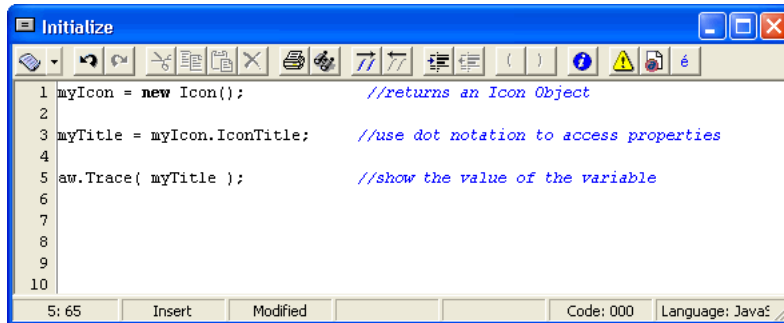


Note: Authorware is selected by default. The status bar shows the language of the calculation icon in the lower right corner of the Calculation window.

JavaScript Variables

Like Authorware Script Language, JavaScript provides global variables. This means that if you define a variable in one icon, the variable's value will be accessible in a later icon.

Here's an example of a variable on first definition (see the definitions of the aw object and the Icon constructor):

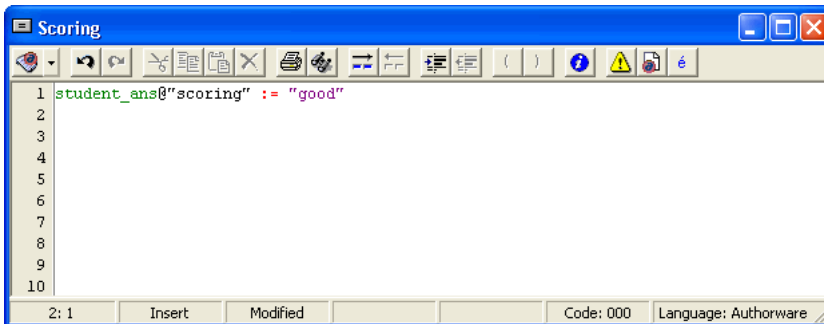


```
1 myIcon = new Icon();           //returns an Icon Object
2
3 myTitle = myIcon.IconTitle;    //use dot notation to access properties
4
5 aw.Trace( myTitle );           //show the value of the variable
6
7
8
9
10
```

5: 65 Insert Modified Code: 000 Language: JavaScript

Authorware Script Language also uses nonglobal variables, called icon variables. Icon variables are defined and accessed similarly to `myvar@"Some_ Icon"`.

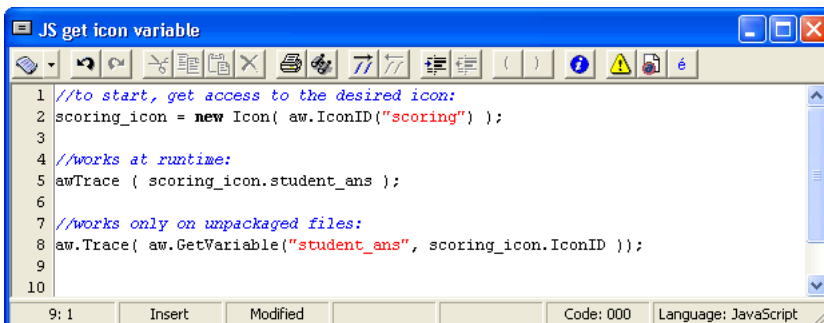
JavaScript in Authorware provides access to icon variables as follows. First, an icon variable is defined and used in Authorware Script Language in one calculation icon:



```
1 student_ans@"scoring" := "good"
2
3
4
5
6
7
8
9
10
```

2: 1 Insert Modified Code: 000 Language: Authorware

Then, the icon variable can be accessed in two different ways from another calculation icon:



```
1 //to start, get access to the desired icon:
2 scoring_icon = new Icon( aw.IconID("scoring") );
3
4 //works at runtime:
5 awTrace ( scoring_icon.student_ans );
6
7 //works only on unpackaged files:
8 aw.Trace( aw.GetVariable("student_ans", scoring_icon.IconID) );
9
10
```

9: 1 Insert Modified Code: 000 Language: JavaScript

Note: If you plan to access an icon variable from JavaScript, do not give it the same name as another icon property or method (for example, movable, animating, checked). You can define JavaScript and Authorware Script Language variables with the same name, although this might get confusing.

In addition to global and icon variables, all of the JavaScript scope rules have been implemented in Authorware. This means that the value of objects, functions, and variables is dependent upon their location in the source text of a program, as that text is interpreted at runtime.

Variables, functions, and objects defined in one calculation icon are available in any later calculation icon. The following are the only restrictions to this rule:

- Execution has to pass through a calculation icon for the definition or change to a variable, function, or object to take place.
- The scope of that variable, function, or object has to permit access to it.

Execution speed

There is a small amount of additional overhead associated with accessing JavaScript variables from Authorware Script Language and vice versa. When scripting in JavaScript, it is generally faster to use only native JavaScript objects, rather than to manipulate Authorware objects in JavaScript. It is better to first copy an Authorware variable to a JavaScript variable, do all of your string parsing with the JavaScript variable, then copy the results back to the Authorware variable. For example:

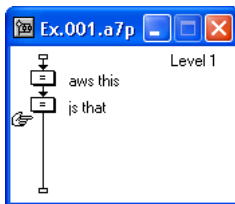
```
var jsString = aw.myString;  
{...manipulation of jsString...}  
aw.myString = jsString;
```

The Calculation editor and languages

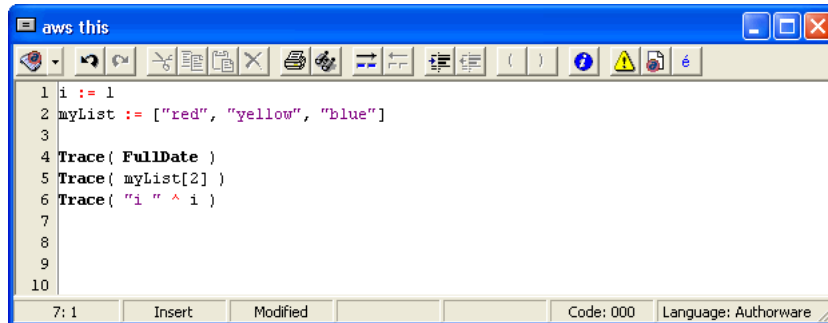
Each calculation icon can be set to either Authorware or JavaScript. The Calculation Editor does not support mixing of languages.

The aw object

All Authorware functions and variables, including user functions and variables, are available to JavaScript by using *aw* notation (use the *aw.* prefix, including the period). This special notation represents an object that Authorware provides to JavaScript. For example:



The first calculation icon is in Authorware Script Language:

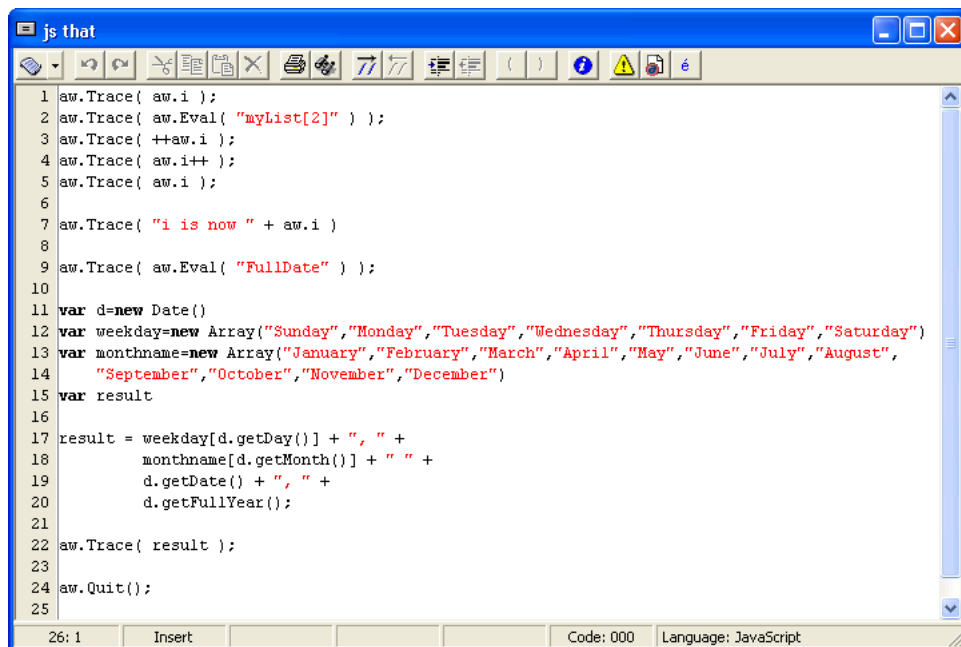


The screenshot shows a window titled "aws this" with a blue title bar and standard Windows window controls. Below the title bar is a toolbar with various icons. The main area contains the following code:

```
1 i := 1
2 myList := ["red", "yellow", "blue"]
3
4 Trace( FullDate )
5 Trace( myList[2] )
6 Trace( "i " ^ i )
7
8
9
10
```

At the bottom of the window, there is a status bar with the text "7: 1", "Insert", "Modified", "Code: 000", and "Language: Authorware".

The second calculation icon is in JavaScript:

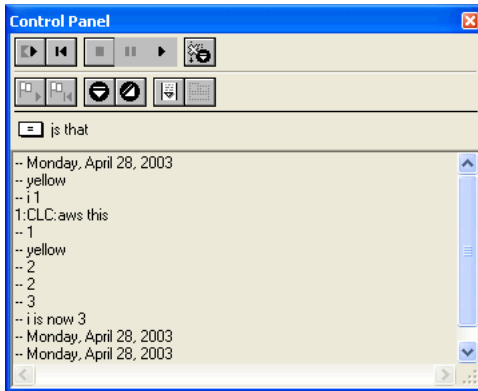


The screenshot shows a window titled "js that" with a blue title bar and standard Windows window controls. Below the title bar is a toolbar with various icons. The main area contains the following code:

```
1 aw.Trace( aw.i );
2 aw.Trace( aw.Eval( "myList[2]" ) );
3 aw.Trace( ++aw.i );
4 aw.Trace( aw.i++ );
5 aw.Trace( aw.i );
6
7 aw.Trace( "i is now " + aw.i )
8
9 aw.Trace( aw.Eval( "FullDate" ) );
10
11 var d=new Date()
12 var weekday=new Array("Sunday","Monday","Tuesday","Wednesday","Thursday","Friday","Saturday")
13 var monthname=new Array("January","February","March","April","May","June","July","August",
14     "September","October","November","December")
15 var result
16
17 result = weekday[d.getDay()] + ", " +
18     monthname[d.getMonth()] + " " +
19     d.getDate() + ", " +
20     d.getFullYear();
21
22 aw.Trace( result );
23
24 aw.Quit();
25
```

At the bottom of the window, there is a status bar with the text "26: 1", "Insert", "Code: 000", and "Language: JavaScript".

When you run it, the Trace window should display the current date:



Case sensitivity

Unlike Authorware Script Language (AWS), JavaScript (JS) is case-sensitive. The function `aw.Trace()` does exist in JS, but the following functions do not exist in JS: `AW.Trace()`, `Aw.Trace()`, `aw.trace()`. Also, for example, the variable “foo” is not the same as “Foo” or “FOO”.

Duplicate system functions and variables

Some Authorware system functions and variables have the same names, for example `IconID` and `IconID()`. JavaScript does not support the use of duplicate function and variable names, so only the function versions are accessible. The following names are the same for both system functions and system variables in Authorware, and exist as functions only in JavaScript:

- `CharCount`
- `CMIReadComplete`
- `Date`
- `Day`
- `DayName`
- `FullDate`
- `IconID`
- `IconTitle`
- `Month`
- `MonthName`
- `NumCount`
- `WordCount`
- `Year`

Note: All of the user functions loaded from a DLL into the file are automatically created as part of the `aw` object.

Icon object

JavaScript code can access icons via the icon object, `new Icon(IconID)`. If `IconID` is the ID of an existing icon, a connection to the icon is created and takes on the properties of the existing icon. If the user passes zero or a nonvalid `IconID`, the currently executing ID is used. This constructor creates a JavaScript object that connects to or mirrors the corresponding Authorware icon; it does not create a new Authorware icon.

The icon object created contains the user icon variables and system `@` variables as properties of the icon object.

For example, if you have a user icon variable called `foo` that is attached to an icon called `myicon`, in JavaScript you would create a new icon object, and then access the value of `foo`:

```
var myicon = new Icon(iconID of myicon);  
myicon.foo = "this is the new value for foo";
```

Now `foo@"myicon"` has a new value, both in JavaScript and as part of the Authorware environment. The two scripting languages are tied to the one pool of data.

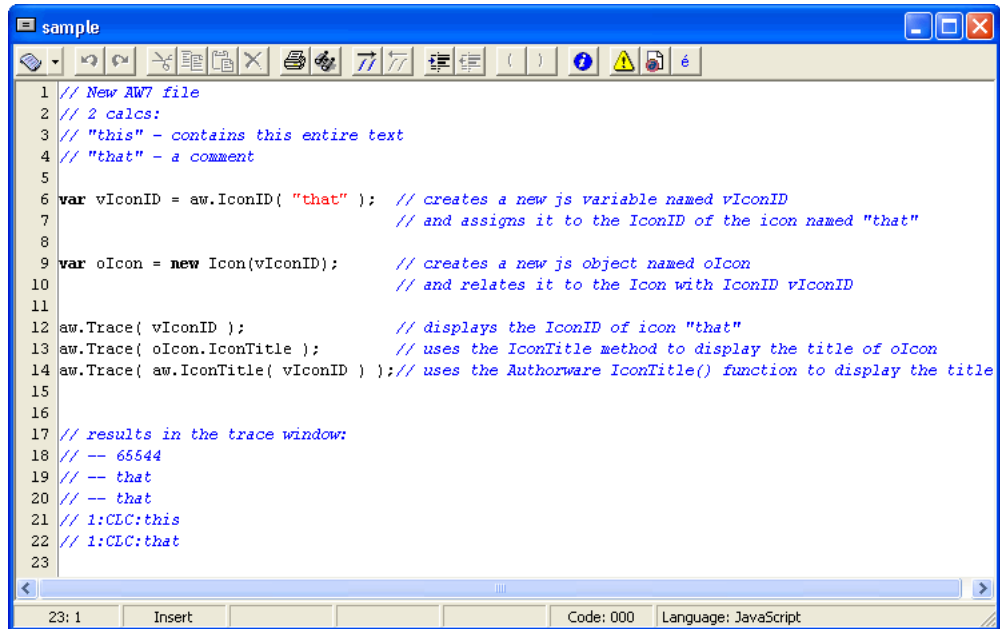
The following methods are available using the icon object.

- ChildNumToID
- DisplayIcon
- DisplayIconNoErase
- EraseIcon
- IconFirstChild
- IconLastChild
- IconNext
- IconNumChildren
- IconParent
- IconPrev
- IconTitleShort
- IconType
- CallScriptIcon
- CallSprite
- CallIcon
- GetCalc
- GetIconContents
- GetIconProperty
- GetSpriteProperty
- SelectIcon
- SetCalc
- SetIconProperty
- SetIconTitle

- SetSpriteProperty
- ReplaceSelection
- Unload

Note: These methods all take one *less* argument than their Authorware object counterparts when used with the icon object. The `IconID` parameter that you would normally pass in each is assumed to be the `IconID` of the icon object.

The following is an example of the code used for these methods:



```

1 // New AW7 file
2 // 2 calcs:
3 // "this" - contains this entire text
4 // "that" - a comment
5
6 var vIconID = aw.IconID( "that" ); // creates a new js variable named vIconID
7                                     // and assigns it to the IconID of the icon named "that"
8
9 var oIcon = new Icon(vIconID);      // creates a new js object named oIcon
10                                    // and relates it to the Icon with IconID vIconID
11
12 aw.Trace( vIconID );               // displays the IconID of icon "that"
13 aw.Trace( oIcon.IconTitle );       // uses the IconTitle method to display the title of oIcon
14 aw.Trace( aw.IconTitle( vIconID ) ); // uses the Authorware IconTitle() function to display the title
15
16
17 // results in the trace window:
18 // -- 65544
19 // -- that
20 // -- that
21 // 1:CLC:this
22 // 1:CLC:that
23

```

New Authorware functions

The following new JavaScript-related functions have been added to Authorware 7:

- EvalJS
- EvalJSFile
- JSGarbageCollect

EvalJS

Syntax

```
result := EvalJS( "expression" )
```

Description

`EvalJS` takes the string in *expression*, interprets it as JavaScript code, and places the resulting value in *result*. This function makes it possible to evaluate JavaScript expressions at runtime, based on information the user enters. Because `EvalJS` can interpret character strings in addition to mathematical expressions, you can evaluate any text the user enters.

`EvalJS` can evaluate expressions that contain the assignment operator (=). This means that you do not need the JavaScript equivalent of `EvalAssign`.

If a syntax error occurred, the system variables *EvalStatus* and *EvalMessage* contain a syntax error message.

Example

To show the contents of a JavaScript *myVar* variable in a display icon, enter the following into the on-screen text:

```
{EvalJS("myVar")}
```

EvalJSFile

Syntax

```
result := EvalJSFile( "filename" )
```

Description

`EvalJSFile` reads the content of the specified filename and executes it as JavaScript code.

JSGarbageCollect()

Syntax

```
JSGarbageCollect()
```

Description

`JSGarbageCollect` immediately performs garbage collection in the JavaScript memory pool. Garbage collection frees space in the memory pool so that the JavaScript engine can reuse it. In general, you should never need to manually call `JSGarbageCollect`, as the JavaScript engine automatically handles garbage collection.

`JSGarbageCollect` performs garbage collection, if necessary, of JavaScript objects, doubles, and strings that are no longer needed by a script. For example, in command scripts that can traverse an entire course file of icons and therefore construct many icon objects, you can call `JSGarbageCollect` to keep memory usage down. The trade-off is that this might reduce the speed of your algorithm.

Enhanced Authorware functions

The following are enhanced Authorware functions:

- SetCalc
- GetIconProperty
- SetIconProperty

SetCalc

Syntax

```
SetCalc(IconID@"IconTitle", "calculation" [,language])
```

Description

The SetCalc function now takes an optional parameter that sets the language of the calculation icon. If the specified calculation string is valid, SetCalc replaces the contents of the specified calculation icon with the specified string. SetCalc sets *EvalStatus* after compiling the specified calculation string. This function only operates on unpackaged files.

SetCalc(IconID@"IconTitle", "calculation") sets the icon language to Authorware Script.

SetCalc(IconID@"IconTitle", "calculation", false) sets the icon language to Authorware Script.

SetCalc(IconID@"IconTitle", "calculation", true) sets the icon language to JavaScript.

GetIconProperty

Syntax

```
GetIconProperty( icon id, #property)
```

Description

The property #awCalcIsJavascript has been added to the GetIconProperty function and returns true or false. This function only operates on unpackaged files.

SetIconProperty

Syntax

```
SetIconProperty(icon id, #property)
```

Description

The property #awCalcIsJavascript has been added to the SetIconProperty function and accepts true or false. This function only operates on unpackaged files.

Helpful hints for using JavaScript

Following are comments, observations, and suggestions about using JavaScript in Authorware 7.

Syntax checking

Authorware provides programmers working with the Authorware Script Language immediate feedback regarding the correct syntax (grammar or construction) of scripts.

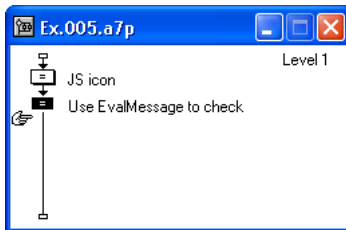
You also get feedback about correct JavaScript syntax, but only at runtime.

The following are two examples of ways to rapidly check JavaScript syntax.

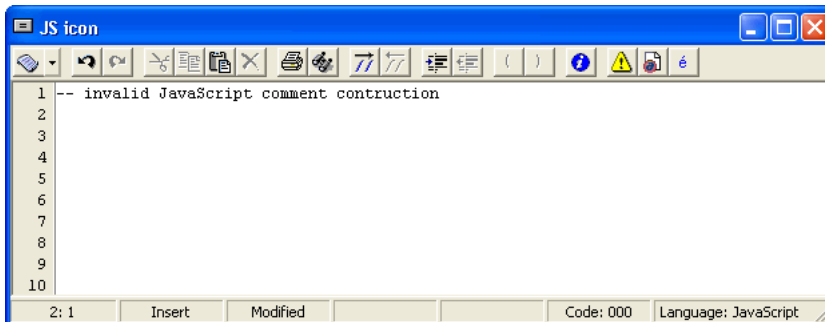
JS syntax checking: example 1

One way to find JavaScript syntax errors uses the Authorware *EvalMessage* system variable. You place an additional icon on the flowline that displays the value of *EvalMessage* to the Trace window.

Consider the following small application:

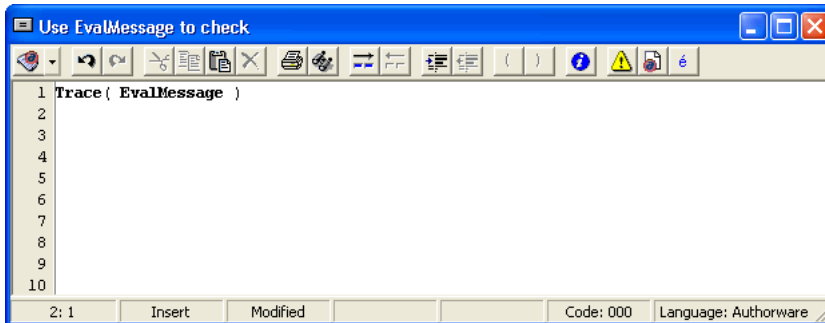


The first calculation icon is set to the JavaScript language and contains a syntax error:

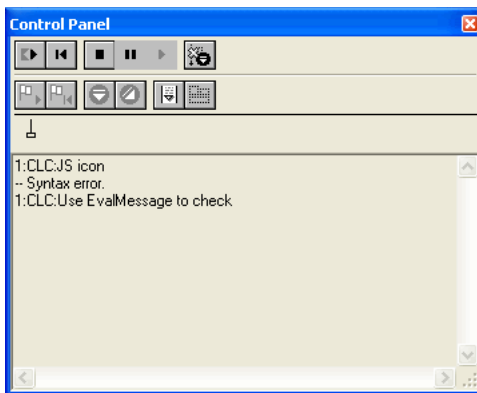


The error is use of the “--” operator without a variable. This operator marks the beginning of a comment in Authorware Script, but is the unary decrement operator in JavaScript.

The second calculation icon, a trace calculation icon, is in Authorware Script Language and simply displays the value of the system variable *EvalMessage* in the Trace window:

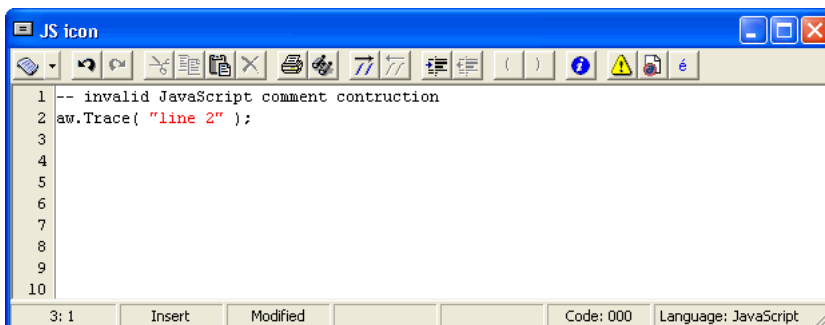


If you open the Trace window and run the application, you get a result like the following:

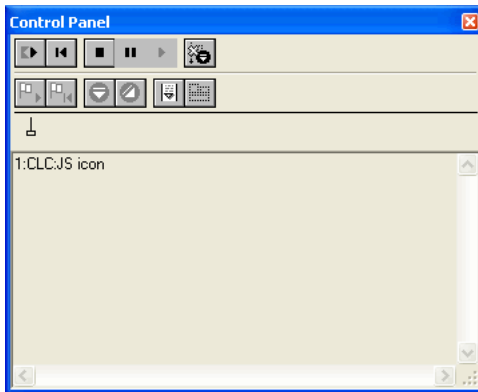


JS syntax checking: example 2

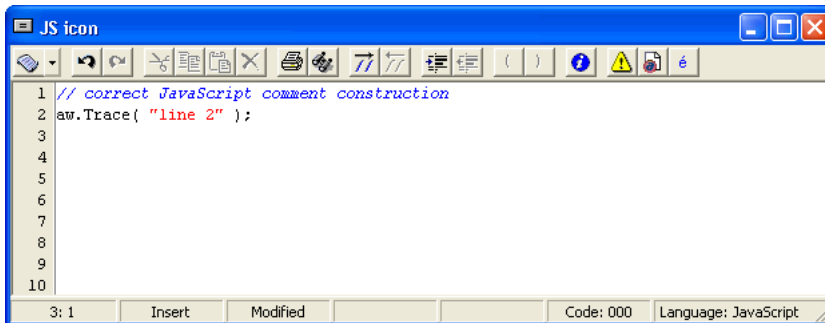
Place the following `aw.Trace()` function in your JavaScript calculation icon:



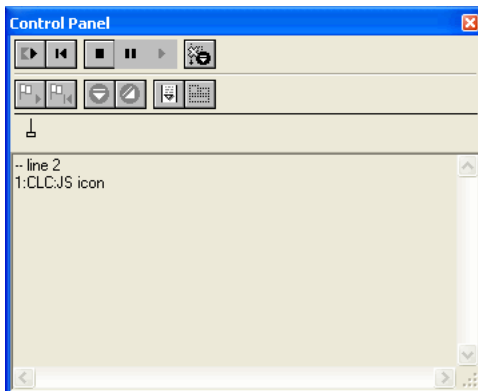
If you open the Trace window and run the application, you get a result like the following:



Correct the error as follows:

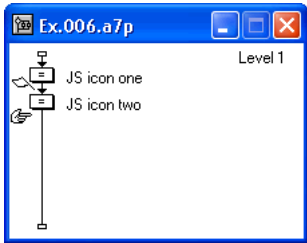


If you open the Trace window and run the application, you get a result like the following:



Using the Start flag

Consider the following simple application comprising two calculation icons, with the Start flag positioned between them, and the Trace window open:



In JS icon one, enter the following JavaScript code:

```
i = "one"
```

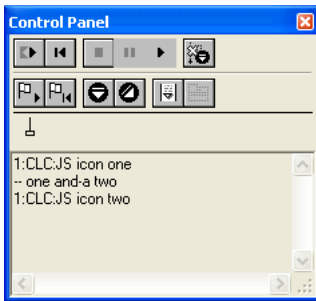
In JS icon two, enter the following JavaScript code:

```
i += " and-a two";  
aw.Trace( i );
```

The first icon creates a global JavaScript variable named *i* and stores the string “one” in it. The second icon appends to *i* the text “ and-a two” and then displays *i* in the Trace window.

Run from the top

If you run the application from the top (Select Control > Restart, or press Control+R), the Trace window appears:



Run from flag with Presentation window open

If you immediately use run from flag (Select Control > Restart from Flag, or Control+Alt+R) without closing the Presentation window, the Trace window appears.

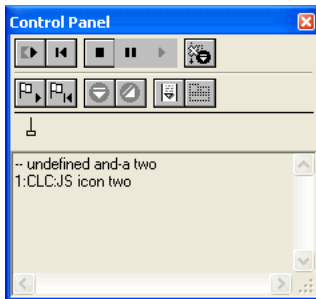
Although this is a simple example, it shows that the value of the variable *i* was retained from the previous execution, and reused correctly.



Run from flag after closing the Presentation window

To run the application from flag, after closing the Presentation window:

- 1 Run the application from the top (Select Control > Restart, or press Control+R).
- 2 Close the Presentation window (Select Window > Presentation, or press Control+1).
- 3 Run from flag (Select Control > Restart from Flag, or press Control+Alt+R).



This time, the value of the variable *i* was reset to “undefined”.

The value of *i* was lost because the JavaScript interpreter was reset when the Presentation window was closed and reopened. If you want to pause and resume an application with JavaScript code, make sure you do not close the Presentation window. If you close the Presentation window, all of the variable values are reset, and you must restart the application from the top.

Using icon titles and variable names

Authorware does not update the names of icons and variables in JavaScript language scripts.

If you refer to an icon by title, as in `aw.IconID("JS icon one")` for example, and then change the icon title, your script will probably be broken.

Likewise, if you refer to an Authorware variable by name, as in `jsvar = aw.Eval("i")`, and then change the variable's name, your script will probably be broken.

JavaScript and the functions and variables windows

JavaScript functions and variables do not appear in the Authorware Functions or Variables windows. This is because JavaScript can construct any variable or function at runtime.

Using semicolons

Terminating JavaScript command lines with semicolons is optional. However, Macromedia recommends using them.

JavaScript and using Resume

With Authorware Script Language, the values of all variables are saved and restored when you set a file's On Return property to Resume. Because JavaScript allows you to create new variables, functions, and objects on the fly, the Resume functionality is not present in this version.

INDEX

Symbols

- + (correct response indicator) 187
- (incorrect response indicator) 187
- (wrong response indicator) 187
- [] (brackets), for optional arguments 169

A

- A7w_data, packaging with 53
- A7WDATA folder, saving user responses in 192
- ActiveX controls
 - CallSprite function with 180
 - for Director Shockwave movies 95
 - functions for 180
 - overview of 182
 - properties, events, and methods of 179
- ActiveX Xtra
 - for web access 210
- AllCorrectMatched variable
 - for tracking performance 189
 - in conditional responses 128
- Anywhere links, going to specific pages with 144
- AppendExtFile function
 - for saving user responses 192
- Application Knowledge Object 38
- archive files, creating 44
- arguments
 - in function syntax 169
 - required vs. optional 169
- Assessment Knowledge Objects 39
- assignment operator (:=)
 - for storing user responses 191
- attributes
 - text 63
- Audio On check box, for digital movies 97
- audio, synchronizing with video and graphics 93

- authoring
 - defined 23
 - good practices in 43
 - icon-based 25
 - stages of 24
 - step-by-step instructions for 25
 - troubleshooting methods for 45
- Authorware
 - books on 15
 - discussion group 15
 - help for 14
 - installing 14
 - naming scheme for files 43
 - Show Me examples for 15, 223
 - step-by-step instructions for using 25
 - user groups for 15
- Authorware Advanced Streamer
 - overview 210
- Authorware piece
 - packaging 52
 - packaging for the web 55
 - preparing for Internet 54
- Authorware TechNotes 15
- Authorware Web Packager
 - font issues with 61
- Authorware Web Player
 - components of 202
 - configuring server for web-packaged files 208
 - downloads and 203
 - font issues with 61
 - functions for use with 206
 - providing to users 209
 - rhythm and pace concerns 44
 - saving user responses with 192
 - Security Knowledge Object 41
 - variables for use with 206

Automatic setting, for conditional responses 126
AVI movies, support for 96
AWARE discussion list 15

B

backing up, methods for 44
Bandwidth function 206
bit depth, for images and graphic objects 87
bitmap resolution, for images and graphic objects 87
bitmap size in web-packaged pieces 203
bitmapped text, role of 62
box
 Box function 90
brackets ([]), for optional arguments 169
branching
 based on calculations 136
 based on conditional expressions 136
 based on expressions 135
 based on variables 135
 decision icons and 134
 random branching 135
 sequential branching 134
 troubleshooting 46
 types of 134
Bring to Front command
 for layering objects 89
Browse Folder Dialog Box Knowledge Object 40
browsers
 distribution options and 51
 testing piece for Internet with 25
Button Editor 118
button library
 description of 118
 media library vs. 107
buttons
 as interaction method 118
 creating 118
 editing 118
 limiting use of 130
 response type symbols and 114
 selecting 118
 sharing as models 109
 states of 118
 step-by-step instructions for 119
 types of 118

C

calculation icons
 comments with 44
 in libraries 106
 role of 28
Calculation window
 comments in 44
 methods and 180
calculations
 branching based on 136
CallIcon function
 for method 180
CallObject function
 for child object 181
CallParentObject function 182
calls
 as round-trip navigation structures 146
 nesting 147
CallSprite function
 for method 180
CallTarget function
 controlling targets with 221
Capitalize function
 controlling text with 64
 syntax example 169
CapsLock variable
 controlling text with 63
CD, as distribution option 51
CD-web hybrid
 advantages of 212
 as distribution option 51, 52, 203
CEL movies
 support for 96
Char function
 controlling text with 64
character variables 167
CharCount function
 controlling text with 64
CharCount variable
 controlling text with 63
check boxes
 setting up 118
Checkboxes Knowledge Object 40
child objects
 description of 181
 functions for 182
 functions for using 182
 identifying 182
 using 181
ChoiceNumber variable 189

- ChoicesMatched variable 189
 - Circle function 90
 - CMI
 - functions for 196
 - variables for 195
 - CMI system
 - setting up courses for 193
 - CMI tab 194
 - Knowledge track settings and 194
 - Code function
 - controlling text with 64
 - Coloring icons 35
 - commands
 - on pull-down menus
 - See also* individual commands
 - Commands menu, adding a command 58
 - comments
 - in authoring practice 44
 - uses for 165
 - compacting files 44
 - compressing files
 - for distribution 52
 - sound files 93
 - conditional expressions, branching based on 136
 - conditional interactions 126
 - conditional user responses
 - Automatic setting for 126
 - step-by-step instructions for 128
 - conditions
 - multiple, checking for 175
 - using 176
 - conditions in scripts
 - if...then statements 175
 - repeat loops 176
 - content
 - creating 59
 - embedding 60
 - incorporating 59
 - linking 60
 - linking to pieces 35
 - management advantages 102
 - management tools 103
 - See also* specific types
 - managing 102
 - media libraries 104
 - OLE used for 59, 86
 - web-packaged pieces and 208
 - Xtra extensions used for 59, 86
 - Continue branching option, in conditional response 128
 - Control Panel
 - for running pieces 24, 37
 - window, troubleshooting with 46
 - conventional DLLs 170
 - Copy File Knowledge Object 40
 - copying files 40
 - correct responses, indicating 187
 - CorrectChoicesMatched variable 190
 - counter, using 176
 - courses
 - setting up for CMI systems 193
 - Create RTF Object Knowledge Object 41
 - creating
 - digital movies 96
 - graphics 88
 - sounds 93
 - text 62
 - video 99
 - cross-platform
 - conversion 96
 - custom functions
 - in DLLs, loading 170
 - in DLLs, using 169
 - methods vs. 180
 - step-by-step instructions for 171
 - custom variables
 - overview of 166
 - saving performance information with 191
 - using 165
 - customizing icons 36
- ## D
- data
 - variables, in authoring development stage 25
 - databases
 - connecting with 200
 - managing information with 192
 - text from 61
 - decision icons
 - branching and 134
 - dragging files onto 35
 - role of 28
 - troubleshooting 45
 - decision structures
 - description of 131, 133
 - decompression program, in distribution package 57
 - DeleteLine function
 - controlling text with 64
 - DeleteObject function
 - for child object 181, 182

- design
 - as authoring stage 24
 - for Internet services 210
 - web-packaged piece 203
- Design window
 - flowline in 23
- development
 - models used in 108
 - using External Media Browser in 105
- diagonal line tool 88
- DIB
 - file support 96
- digital movie
 - functions 98
 - functions *See also* specific functions
 - icons, in libraries 106
 - icons, role of 28, 96
 - variables, listed 98
- digital movie variables
 - See also* specific variables
- Direct to Grid motion type 154
- Direct to Line motion type 154
- Direct to Point motion type 154
- directing flow
 - automatically 133
 - overview of 131
- Director movies
 - importing 96
 - in distribution package 57
 - retaining interactivity of 97
 - saving as QuickTime movies 97
 - support for 96
- Director Shockwave movies, Shockwave ActiveX
 - control for 95
- DirectToScreen variable
 - for images 90
- discussion group, Authorware 15
- disk space
 - distribution options and 44
- disk, as distribution option 51
- disk-and-network delivery, as distribution option 51, 52
- display icons
 - in libraries 106
 - rearranging order with 89
 - role of 28, 87
 - troubleshooting 45
- DisplayHeight variable
 - for images 90
- DisplayLeft variable 90
- DisplayTop variable 90
- DisplayWidth variable 90
- distribution
 - basics of 50
 - digital movie issues 96
 - external content issues 103
 - intranet options for 201
 - libraries and 107
 - media for 51
 - network options for 201
 - of Authorware Web Player 209
 - of files with Authorware pieces 56
 - optimizing file size for 52
 - World Wide Web options for 201
- DLLs
 - as custom functions 170
 - conventional DLLs 170
 - custom functions in 169
 - search path and 170
- downloading
 - design considerations for 203
 - functions for 206
 - non-web-packaged pieces 210
 - onDemand option 208
- downloads
 - Internet considerations for 149
- Drag and Drop exercises 39
- Drag and Drop Question Assessment Knowledge Object 39
- drag-and-drop feature
 - for importing content 35
 - for importing digital movies 96
 - for importing sound files 93
 - for importing text 63
 - for linking content 35
 - to target area 124
- dragging objects
 - letting users drag objects 155
- DrawBox function 90
- DrawCircle function 91
- drawing
 - creating graphics 88
- DrawLine function 91

E

- editing
 - links 60
 - text 61
- educational applications
 - database connectivity for 200
 - Internet access with 210
- embedding
 - and linking content 60
 - content, description of 59
 - content, linking content vs. 60
 - content, step-by-step instructions for 60
 - content, to flowline 35
- Empty option 220
- EntryText variable
 - controlling text with 63
 - in text-entry user response 123
 - user responses recorded in 191
- erase icons
 - role of 28
- Eval function
 - controlling text with 64
- EvalAssign function
 - controlling text with 64
- evaluation, as authoring stage 25
- EventLastMatched variable 180
- EventQueue variable 180
- events
 - defined 180
 - limiting use of 130
 - overview of 128
 - reacting to 128
 - sprite 179, 180
 - tracking 128
 - variables for managing 180
- EventSuspended variable 180
- exercises, types of 39
- Exit Framework/Return option 144
- Explorer (Windows), dragging or dropping content
 - from 35
- expressions
 - as External Media Browser paths 103, 106
 - branching based on 135
 - defining links with 60
 - in scripts 174
 - overview of 157, 171
 - using 171
 - using to go to pages 145
 - where used 158

- external content
 - features 103
 - naming scheme for 44
 - rhythm and pace concerns 44
 - using 105
 - web-packaged pieces and 208
- external images, linking to 89
- External Media Browser
 - development uses 105
 - distribution uses 106
 - features 106
 - managing content with 103
 - role of 60, 103
 - step-by-step instructions for 106
 - using 105
- external software modules, in distribution package 57
- external sounds, in distribution package 57

F

- feedback
 - providing 190
 - with conditional responses 128
- File command (Modify menu), for setting up file
 - properties 43
- File Knowledge Objects 40
- file properties
 - setting up 43
- File Properties dialog box
 - CMI tab in 194
 - options, listed 43
- File Transfer Protocol (FTP) functions 192
- FileIO Xtra
 - GetOSDirectory function 181
 - version method in 182
- FileLocation variable
 - paths to folders stored in 192
- filenames
 - naming scheme for 43
- files
 - compacting 44
 - dragging onto flowline 35
 - external content 103, 106
 - jumping to 148
 - linking to 35, 89
 - media libraries vs. 104
 - naming scheme for 43, 44
 - network download speed 201
 - optimizing size of 52

- saving 44
- saving user responses in 192
- selecting for distribution 56
- Find CD Drive Knowledge Object 40
- Find command
 - (Edit menu), for text 63
- Find function
 - controlling text with 64
- finding
 - text 63
- FirstTryCorrect variable
 - in performance summary 190
 - tracking and 190
- FirstTryWrong variable
 - in performance summary 190
 - tracking and 190
- FLC/FLI movies
 - support for 96
- flow
 - directing 131
 - directing automatically 133
 - Internet and 132
- flowline
 - breaking into sections 43
 - comments on 44
 - dragging files onto 35
 - embedding or linking content to 35
 - importing digital movies to 96
 - importing sound files to 93
 - in icon-based authoring 23
 - inserting sprites on 178
 - interaction 114
 - models 105, 108
 - troubleshooting methods 45
- FlushEventQueue function 180
- Font Resources Assessment Knowledge Objects 40
- fonts
 - in distribution package 57
- ForceCaps variable
 - controlling text with 63
- formatting, imported text and 62
- framework environment
 - how it works 140
- framework icons
 - dragging files onto 35
 - overview of 140
 - role of 28
- framework window 140
- frameworks
 - description of 139
 - navigation controls for 141
 - nesting navigation frameworks 142
- FTP functions
 - as distribution option 210
 - listed 211
- FTP UCD 211
 - (Windows), storing user responses with 192
- FtpAbort function 211
- FtpAppend function 211
- FtpChangeParentDir function 211
- FtpChangeWorkingDir function 211
- FtpClose function 211
- FtpConnect function 211
- FtpDelete function 211
- FtpDisconnect function 211
- FtpGetWorkingDir function 211
- FtpList function 211
- FtpMakeDir function 211
- FtpNameList function 212
- FtpOpen function 212
- FtpRemoveDir function 212
- FtpRename function 212
- FtpResult function 212
- FtpRetrieve function 212
- FtpStatus function 212
- FtpStore function 212
- functions
 - CMI functions 196
 - controlling images with 90
 - controlling sound with 94
 - controlling video with 100
 - custom, methods vs. 180
 - digital movie 98
 - for event management 180
 - for Knowledge Objects 216
 - for movies 98
 - for parent and child objects 182
 - for sprite icons and properties 179
 - for web-packaged pieces 206
 - image, listed 90
 - in scripts 174
 - Internet-related 206
 - listed 211
 - overview of 157, 168
 - purpose of 158
 - sound, listed 95
 - step-by-step instructions for 165, 171
 - syntax of 169

- system, overview of 168
- text, listed 64
- troubleshooting 46
- video, listed 101
- where used 158

G

- Get RTF Object Text Range Knowledge Object 41
- GetIconProperty function
 - for sprite icons 179
- GetLine function
 - controlling text with 64
- GetMovieInstance function 98
- GetNumber function
 - controlling text with 64
- GetSpriteProperty function 179
- GetWord function
 - controlling text with 64
- global functions
 - identifying 182
 - using 181
- GlobalPreroll variable 206
- GlobalTempo variable 180
- Go Back links 144
- GoToNetPage function 207
- graphics
 - as content 59
 - controlling with functions 90
 - controlling with variables 90
 - creating 88
 - embedding and linking 60
 - integrating 88
 - positioning 89
 - step-by-step instructions for 91
 - synchronizing with audio and video 93
 - working with 87
- graphics file formats, Internet delivery and 203
- graphics files, in distribution package 57
- graphics functions
 - listed 90
 - See also* specific functions
- graphics variables
 - listed 90
 - See also* specific variables
- grid, displaying images on 89
- Group command
 - for selected icons 35

- grouping
 - icons 35
 - images 89
- Guidelines for using sound 92

H

- hard disk capacity, distribution options and 44
- hard disk, saving user responses on 192
- hardware requirements
 - bitmap resolution and 87
 - distribution options and 50
 - fonts and 61
- help features
 - Authorware Attain Help Pages 14
 - for sprite Xtras 179
 - step-by-step guide 25
- Hot Area exercises 39
- Hot Object exercises 39
- Hot Object Question Knowledge Object 39
- hot objects
 - limiting use of 130
 - overview of 121
 - step-by-step instructions for 122
- Hot Spot Question Knowledge Object 39
- hot spots
 - limiting use of 130
 - overview of 120
 - response type symbols and 114
 - step-by-step instructions for 121
 - working with 120
- hot text
 - attaching navigation links to 148
 - role of 61
- HotTextClicked variable 189
 - controlling text with 64
- hypertext
 - creating 148
 - creating links 148
 - HotTextClicked variable and 189
 - navigating with 148
 - role of 61
 - step-by-step instructions for 148

I

Icon color palette

role of 29

Icon palette

description of 23

icon properties

troubleshooting 46

icon titles

comments in 44

icon-based authoring 25

icons

coloring 35

comments with 44

customizing 36

embedding or linking content to 35

framework icons 140

grouping in a map 35

in library files 104

Knowledge Object icons 220

motion icons 154

naming 35

naming scheme for 43

navigate icons 139

rearranging 35

result icons 113

selecting multiple 35

troubleshooting methods 45, 46, 47

working with on flowline 44

See also Knowledge Object icon

If...then statements

in scripts 175

image file formats, Internet delivery and 203

image functions, listed 90

See also specific functions

Image Properties command

authoring and 35

image variables

See also specific variables

image variables, listed 90

images

bit-depth considerations 87

bitmap resolution for 87

combining sound with 93

controlling with functions 90

controlling with variables 90

creating 88

dragging onto display icon 35

importing, platform and 88

integrating 88

Internet delivery issues 203

linking to 89

positioning 89

properties of 35

step-by-step instructions for 91

synchronizing sound with 93

working with 87

implementation, as authoring stage 25

Import command

in icon-based authoring 35

importing

graphics, bitmap resolution and 87

movies, quick method for 96

sounds, description of 93

text, uses for 62

types of images 88

In Area property 153

incorrect responses, indicating 187

Insert RTF Object Hot Text Interaction Knowledge

Object 41

InsertLine function

controlling text with 64

Inspectors command (Windows menu), for images 89

installation instructions, distribution options and 51

installer program

distribution options and 51

in distribution package 57

installing

Authorware 14

interaction flowline 114

interaction icons 112

dragging files onto 35

graphics in 87

how they work 112

in libraries 106

role of 28

Interaction Properties dialog box 194

interactions

advantages of 111

components of 111

conditional 126

in authoring development stage 25

Internet delivery issues 203

judging 187

limiting 130

methods 116

See also specific methods

overview of 111

results of 112

saving as models 105

step-by-step instructions for 117

- structure of 112
- text entry by users 123
- time limits and 126
- tracing flow through 114
- using 111
- Interface Components Knowledge Objects 40
- Internet
 - applications, packaging and distributing 25
 - as distribution option 51–??, 52, ??–52, 201
 - bit-depth considerations 87
 - CD-web hybrid solutions 203, 212
 - flow and 132
 - functions for 206
 - jumping to other files and 149
 - navigation structures and 142
 - preparing a piece for 54
 - saving user responses over 192
 - services, designing for 210
 - services, using 210
 - speed considerations 201
 - variables for 206
- Internet Knowledge Objects 41
- intranet
 - applications, speed considerations 201
 - as distribution option 52
 - Authorware Web Player and 201
 - packaging and distributing applications 25
 - web access and applications 210

J

- JudgedInteractions variable
 - in performance summary 190
 - tracking and 190
- JudgedResponses variable
 - in performance summary 190
 - tracking and 190
- judging interactions 187
- Jump to Authorware File Knowledge Object 40
- JumpFile function 148, 206
 - URLs with 211
- JumpFileReturn function 207
 - URLs with 211
- jumps
 - as one-way navigation structures 146
 - Internet considerations for 149
 - navigation links and 144
 - to files 148
 - to specific pages 144

K

- keypress responses
 - limiting use of 130
 - overview of 122
 - using 122
- keywords
 - naming scheme for 43
 - using 44
- Knowledge Object icons
 - description of 220
 - options for 220
 - properties for 220
- Knowledge Objects 38
 - communicating with 221
 - displaying 214
 - for a new file 37
 - functions for 216
 - introduction to 37
 - launching wizards from 215
 - overview for developers 213
 - step-by-step instructions for 219
 - uses for 213
 - using 38
 - variables for 215
 - wizard user interface and 220
- Knowledge Objects window 214
- Knowledge Track
 - overview of 193
 - settings for 193

L

- LastLineClicked variable 190
- LastObjectClicked variable 190
- LastParagraphClick variable 190
- LastWordClicked variable 190
- LastX/LastY variables 90
- Launch Default Browser Knowledge Object 41
- Layer setting
 - for digital movies 97
 - for display icon 90
- Layer variable 90
- layering text and graphics 89
- libraries
 - adding icons to 106
 - button 118
 - in Authorware 104
 - packaging 53
 - step-by-step instructions for 108
 - using 106

- library search path, as distribution requirement 51
- limiting user responses 130
- Line function 91
- LineClicked variable 189
- LineCount function 64
 - controlling text with 64
- Link to File option
 - for linking images 89
- linked files, in distribution package 57
- linking content 60
 - digital movies 96
 - embedding content vs. 60
 - external content files 103
 - external images 89
 - External Media Browser for 103
 - media libraries 104
 - models 105
 - step-by-step instructions for 60
- linking content to flowline
 - dragging from browser and 35
 - File menu, Import command, in icon-based authoring 35
- links
 - editing 60
 - navigation links 144
- List Recent Pages links 144
- lists
 - using 176
- local area network (LAN)
 - as distribution medium 51
 - saving user responses on 192
- Lock Icon option 220
- logical variables, for conditional responses 127
- Login for an assessment 39
- Login Knowledge Object 39
- LowerCase function
 - controlling text with 64

M

- Macromedia Press 15
- Macromedia website, providing link to 209
- map icons
 - as result icon 112, 113
 - role of 28
- MapChars function
 - controlling text with 64
- MatchedEver variable 190
- MatchedIconTitle variable 190

- media
 - available types of 59
 - distribution options 51
 - See also* external media
 - synchronization 36, 94
- media editors 59
- media libraries 104
 - as design tool 24
 - button libraries vs. 107
 - features 107
 - step-by-step instructions for 108
 - using 106
- media synchronization 94
- MediaLength variable, for digital movie 98
- MediaPause function
 - for digital movie 98
 - for sound 95
- MediaPlay function
 - for digital movie 98
 - for sound 95
- MediaPlaying variable
 - for digital movie 98
 - for sound 94
- MediaPosition variable
 - for digital movie 98
 - for sound 94
- MediaRate variable
 - for digital movie 98
 - for sound 94
- MediaSeek function
 - for digital movie 98
 - for sound 95
 - for video 101, 102
- menus
 - pull-down 119
 - pull-down, limiting use of 130
- Message Box Knowledge Object 40
- methods
 - custom functions vs. 180
 - defined 180
 - parent and child objects and 181
 - sprite 180
 - sprite, properties 179
- Microsoft Calendar ActiveX control, events sent by 180
- Microsoft Web Browser, events sent by 180
- Microsoft Word, RTF text in 62
- MidXtra, child object created from 181
- minus sign (-)
 - as incorrect response indicator 187

- models
 - defined 108
 - features 109
 - sharing 109
 - step-by-step instructions for 109
 - uses for 105
 - using 108
 - motion effects
 - description of 151
 - in authoring development stage 25
 - tracking for objects 155
 - with digital movies 97
 - working with 154
 - motion icons
 - role of 28
 - using 154
 - motion types
 - Direct to Grid 154
 - Direct to Line 154
 - Direct to Point 154
 - Path to End 155
 - Path to Point 155
 - Move Cursor Knowledge Object 40
 - Movie Controller Knowledge Object 40
 - Movie Icon Properties dialog box, settings in 97
 - movie synchronization with display icons 94
 - movies
 - as content 59
 - combining sound and graphics in 94
 - creating 96
 - distribution issues 96
 - embedding and linking 60
 - formats for 96
 - functions for 98
 - integrating 96
 - media synchronization 94
 - on asset list 24
 - playing range of frames from 97
 - positioning 89
 - step-by-step instructions for 98
 - using efficiently 97
 - using functions with 98
 - using motion with 97
 - using variables with 98
 - variables for 98
 - working with 95
 - moving objects
 - with motion icon 154
 - MPEG file support, for movies 96
 - multimedia pieces
 - authoring help guide 25
 - authoring practice tips 43
 - authoring stages 24
 - creating content for 59
 - icons in 23, 28–29
 - interaction for. *See* interactions
 - packaging 56
 - troubleshooting 45
 - Multiple Choice exercises 39
 - Multiple Choice Question Assessment Knowledge Object 39
 - multiple icons, selecting 35
- N**
- naming icons 35
 - naming scheme, creating 43–44
 - navigate icons
 - description of 139
 - role of 28
 - navigating
 - with hypertext 148
 - navigation controls
 - for frameworks 141
 - navigation frameworks
 - nesting 142
 - navigation links
 - attaching to hot text 148
 - step-by-step instructions for 147
 - using navigate icons for 144
 - working with 144
 - navigation structures
 - considerations for 142
 - description of 131, 139
 - in authoring development stage 25
 - Internet and 142
 - overview 139
 - step-by-step instructions for 143
 - Nearby links 144
 - nested repeat loops, in scripts 174
 - nesting
 - calls 147
 - navigation frameworks 142
 - NetAbort function 207
 - NetBrowserName variable 206
 - NetBrowserVendor variable 206
 - NetBrowserVersion variable 206
 - NetConnected variable 206
 - NetDownload function 207
 - NetDownloadBackground function 207

- NetDownloadName function 207
- NetError function 207
- NetFileSize function 207
- NetLastModDate function 207
- NetLocation variable 206
- NetPercentDone function 207
- NetPreload function 207, 208
- network
 - as distribution option 51, 201
 - disk delivery combined with 51, 52
 - saving user responses on 192
- New File Knowledge Objects 38
- NewObject function
 - for child object 181, 182
- NumCount function
 - controlling text with 64
- NumCount variable
 - for conditional response 127
- NumEntry/NumEntry2/NumEntry3 variables
 - user responses recorded in 191
- numerical
 - variables 167

O

- object type, icon type and 23
- object-oriented programming, parent and child objects 181
- ObjectClicked variable 189
- ObjectMatched variable 189
- ObjectMoved variable 189
- objects
 - hot 122
 - layering 89
 - letting users drag 155
 - moving 154
 - on asset list 24
 - positioning 152
 - positioning on a path 153
 - positioning on the screen 153
 - positioning within an area 153
 - tracking motion of 155
 - tracking position of 155
- OLE objects
 - as content 59, 86
- On Path property 153
- onDemand download option
 - content management with 208

- One-button publishing
 - purpose of 202
 - settings 55
 - using to package a piece 55
- one-way trips. *See* jumps
- online help 14
- Open Database Connectivity (ODBC) capabilities
 - advantages of 200
 - sample uses for 61
- Open File Dialog Box Knowledge Object 40
- Open File Dialog Knowledge Object 40
- operating system
 - distribution options and 50
 - testing with 50
- operators
 - overview of 171
 - types of 171
- optimizing file size 52
- oval tool 88
- Overlapping function 91

P

- pace, in authoring practice 44
- packaging
 - basics of 52
 - files included in 53
 - in distribution stage 25
 - libraries 53
 - process for 52
 - step-by-step instructions for 54
- pages
 - backtracking to 144
 - jumping to 144
 - searching for 145
 - using expressions to go to 145
- paging structures
 - description of 137
 - for text 63
 - RTF files and 137
- ParagraphClicked variable 190
- paragraphs
 - ruler 62
- parent objects
 - description of 181
 - functions for using 182
 - identifying 182
 - using 181
- parsing XML files 183
- pasting text 62

- path
 - for web-packaged piece 208
 - in External Media Browser 103, 106
 - to records folder 192
- Path to End motion type 155
- Path to Point motion type 155
- paths
 - positioning objects on 153
- Peachpit Press 15
- PercentCorrect variable
 - in performance summary 190
 - tracking and 190
- PercentWrong variable
 - in performance summary 190
 - tracking and 190
- performance
 - bit depth and 87
 - providing summary 190
 - tracking and recording 185
 - tracking and recording, providing feedback 190
 - tracking and recording, step-by-step instructions for 193
 - tracking and recording, storing information 191
 - tracking and recording, system variables for 188
- perpetual interactions
 - pull-down menus as 119
 - step-by-step instructions for 129
- PICS movies
 - as external files 96
 - See also* movies
- plus sign (+)
 - as correct response indicator 187
- pointer tool
 - for positioning graphics 89
 - using 88
- polygon tool 88
- polygon tool, using 88
- position, tracking for objects 155
- positioning objects
 - description of 151, 152
 - on a path 153
 - on the screen 153
 - within an area 153
- Preload function 207, 211
- Preroll variable 206
- Presentation window
 - hot spots on 120
 - in icon-based authoring 23
 - Internet delivery and 203
 - layering graphic objects in 90
 - sprite Xtras for 178
 - using Text tool in 62
 - using toolbox in 88
- PresetEntry variable, controlling text with 64
- PreviousMatch@IconTitle variable 190
- properties
 - defined 179
 - In Area 153
 - Knowledge Object icons 220
 - of sprite icons 179
 - of sprite icons vs. sprites 179
 - On Path 153
- Properties dialog box
 - for positioning graphics 89
- prototypes, as authoring stage 24
- pull-down menus
 - overview of 119
 - setting up 119
 - step-by-step instructions for 120

Q

- QuickDraw 3D sprite icon, properties of 179
- QuickTime movies and QuickTime for Windows
 - drivers 57
 - in distribution package 57
 - saving Director movies as 97
 - support for 96
 - See also* movies
- quitting
 - wizards 215
- Quiz Knowledge Object 39

R

- radio buttons
 - description of 118
 - uses for 118
- Radio Buttons Knowledge Object 40
- RAM requirements
 - distribution and 50
- random branching
 - overview of 135
- Randomly To Unused Path option 135
- Read INI Value Knowledge Object 40
- Read Me files, distribution options and 51
- ReadExtFile function 207
 - URLs with 211
- ReadURL function 207
- rearranging icons 35
- Recent navigation link 144

- recording performance
 - in Authorware 191
 - in database 192
 - in external files 192
 - providing feedback 190
 - step-by-step instructions for 193
 - system variables for 188
- records folder, saving user responses in 192
- RecordsLocation variable
 - paths to folders stored in 192
- rectangle 88
- Reduce function
 - controlling text with 64
- referral applications, database connectivity for 200
- relative path
 - advantages of 106
- removable disk, as distribution option 51
- repeat loops
 - in scripts 176
- RepeatString function
 - controlling text with 64
- Replace function
 - controlling text with 64
- ReplaceLine function
 - controlling text with 65
- ReplaceString function
 - controlling text with 65
- ReplaceWord function
 - controlling text with 65
- Reset Path On Entry option 135
- ResetBandwidth function 207
- Response Properties Dialog Box 194
- response type symbol
 - in interaction 112, 113
- responses. *See* user responses
- ResponseStatus variable 190
- ResponseTime variable 188
- result icons
 - content of 113
 - interactions and 112
 - response type symbols and 113
- result path 112, 114
- results, in interactions 112
- RFind function
 - controlling text with 65
- RGB function 91
- rich text format (RTF) files
 - importing text in 62
 - paging structure and 137
- right responses, indicating 187

- round trips. *See* calls
- rounded rectangle tool 88
- RTF files
 - paging structure and 137
- RTF functions 83
- RTF Knowledge Objects
 - Create RTF Object Knowledge Object 81, 82
 - Get RTF Object Text Range Knowledge Object 81, 82
 - Insert RTF Object Hot Text Interaction Knowledge Object 81
 - RTF Object Hot Text Interaction Knowledge Object 82
 - Save RTF Object Knowledge Object 81
 - Search RTF Object Knowledge Object 81, 83
 - Show or Hide RTF Object Knowledge Object 81, 82
 - using with RTF document 81
- RTF Objects Editor 66
- RTF Objects Knowledge Objects 41
- RTF Objects, publishing for the web 86
- RTFObj.u32 83
- Run Wizard option 220
- runtime applications 53

S

- sampling rate, for sounds 92
- Save and Compact command
 - File menu 44
- Save As command
 - File menu 44
- Save command
 - File menu 44
- Save File Dialog Box Knowledge Object 40
- Save RTF Object Knowledge Object 41
- saving
 - performance information 191
 - your work 44
- Scoring for an assessment 39
- Scoring Knowledge Object 39
- screen
 - displaying text on 61
 - positioning objects on 153
- scripting 174
 - defined 174
 - if...then statements 175
 - overview of 174
 - repeat loops 176
 - step-by-step instructions for 177

- scripting Xtra extensions
 - overview of 181
 - parent and child objects with 181
- scrolling structure for text 63
- Search navigation links, searching for words and 145
- search path
 - DLLs and 170
- Search RTF Object Knowledge Object 41
- searching
 - for pages 145
 - for text 63
- security features
 - non-web-packaged pieces and 211
 - saving user responses and 192
- selecting multiple icons 35
- Send Mail Knowledge Object 41
- Send to Back command
 - for layering objects 89
- senders
 - overview of 128
 - sprite events and 180
 - sprite methods and 180
- SendEventReply function 180
- sequential branching
 - overview of 134
- server
 - configuring for web-packaged files 208
- Set File Attribute Knowledge Object 40
- Set Window Caption Knowledge Object 40
- SetFill function 91
- SetFrame function 91
- SetIconProperty function
 - for sprite icon 179
- SetLine function 91
- SetMode function 91
- SetPalette function 91
- SetSpriteProperty function 179
- Shockwave ActiveX control, for Director Shockwave
 - movies 95
- Short Answer Question Knowledge Object 39
- Short Text exercises 39
- shortcut commands
 - on pull-down menus
- Show Me examples 15, 223
- Show or Hide RTF Object Knowledge Object 41
- Single Choice Question Knowledge Object 39
- Slider Knowledge Object 40
- software modules, in distribution package 57
- sound
 - distribution options and capabilities 51
 - functions *See also* specific functions
 - functions, listed 95
 - synchronization with display icons 94
- sound files
 - compressing 93
 - creating 93
 - editing before importing 92
 - embedding and linking 60
- sound icons
 - in libraries 106
 - role of 28
- sound variables
 - listed 94
 - See also* specific variables
- SoundBytes variable 206
- SoundPlaying variable 94
- sounds
 - as content 59
 - combining with other media 93
 - considerations for using 92
 - controlling with functions 94
 - controlling with variables 94
 - creating 93
 - file formats for 93
 - integrating 93
 - media synchronization 94
 - sampling rate issues 92
 - step-by-step instructions for 95
 - working with 92
- special
 - effects, using transitions for 151
- speed issues on networks
 - CD-web hybrids and 212
- sprite icons
 - properties of 179
- sprite Xtra extensions
 - overview of 178
- sprite Xtras
 - properties, events, and methods of 179
- sprites
 - properties of 179
 - senders 180
- start and stop flags
 - in troubleshooting 45
 - role of 29
- storing information, on user performance 191
- straight line tool 88

- streaming
 - Authorware Web Player 202
 - content management with 208
 - design considerations for 203
 - providing Authorware Web Player for 209
 - technology overview 202
- String function
 - controlling text with 65
- Strip function
 - controlling text with 65
- Structured Query Language (SQL)
 - advantages of 200
 - sample use for 61
- structures
 - decision structures 133
 - navigation structures 139
 - paging structures 137
- styles, text 63
- subroutines
 - overview 149
- SubStr function
 - controlling text with 65
- synchronizing sound, movies, and display icons 94
- syntax
 - in Functions dialog box 169
 - of functions 169
- system functions
 - overview of 168
 - parent and child object methods vs. 181
- system variables
 - for conditional responses 127, 128
 - for tracking performance 188
 - overview of 166
 - using 165
- SystemSeconds variable 188

T

- target areas
 - limiting use of 130
 - step-by-step instructions for 124
 - using 124
- target responses
 - checking for match with 114, 187
 - definition of 112
- targets
 - controlling with wizards 221
 - description of 214
- TechNotes 15

- testing pieces
 - by running 44
 - for Internet delivery 203
 - in authoring development stage 25, 44
 - in authoring evaluation stage 25
 - on different computers 50
- text
 - as content 59
 - combining sound with 94
 - creating within Authorware 62
 - database, sending to or importing from 61
 - displaying on screen 61
 - dragging onto display icon 35
 - editing 61
 - embedding and linking 60
 - font issues 61, 62
 - functions *See also* specific functions
 - functions, listed 64
 - hypertext 61
 - importing, uses for 62
 - integrating into Authorware piece 62
 - issues concerning 61
 - overview of 61
 - paging or scrolling structure for 63
 - pasting 62
 - searching for 63
 - step-by-step instructions for 65
 - style attributes 63
 - styles, sharing as models 109
 - synchronizing sound with 94
 - variables
 - See also* specific variables
 - variables, listed 63
 - working with 61
- Text menu
 - using 61
- text tool 88
- text-entry fields
 - conditional responses for 127
 - creating 123
 - providing feedback for 127
 - response type symbols and 114
 - step-by-step instructions for 123
- text-entry interactions
 - overview of 123
- time limits for responses 126
 - limiting use of 130
 - step-by-step instructions for 126
- TimeInInteraction variable
 - for user response 126, 188

- TimeRemaining variable 188
 - for user response 126
- TimesMatched variable 190
- tool, using 88
- toolbar
 - description of 27
 - illustration of 23
 - showing or hiding 27
- toolbox
 - using 88
- TotalCorrect variable 190
 - in performance summary 190
- TotalWrong variable 190
 - in performance summary 190
- Trace function
 - in troubleshooting variables 46
- tracking events 128
- tracking performance
 - providing feedback 190
 - step-by-step instructions for 193
 - storing information 191
 - system variables for 188
- transition settings 152
- transition Xtra extensions
 - description of 152
- transitions
 - description of 151
 - using for special effects 151
 - working with 151
- Transitions dialog box 152
- tries limit response
 - description of 125
 - step-by-step instructions for 126
- Tries variable 188
- troubleshooting
 - authoring methods 45
 - interactions 114
 - links 60
 - step-by-step instructions for 47
- True/False exercises 39
- True/False Question Knowledge Object 39

U

- UCDs
 - DLLs and 170
- UpperCase function
 - controlling text with 65
- user groups, Authorware 15

- user responses
 - buttons for 118
 - catch-all areas for 120
 - conditional 128
 - hot objects for 122
 - hot spots for 120
 - in interactions 111, 112
 - limiting 130
 - marking right or wrong 187
 - pull-down menus for 119
 - response type symbols 112, 113
 - target areas for 124
 - text-entry fields for 123
 - time limits for 126
 - tries limits for 126
- UserName variable
 - saving user responses with 192
- users
 - feedback for 190
 - letting users drag objects 155

V

- variables
 - as External Media Browser paths 103, 106
 - branching based on 135
 - character 167
 - CMI variables 195
 - controlling images with 90
 - controlling sound with 94
 - controlling video with 100
 - custom variables 165
 - digital movie
 - for event management 180
 - for Knowledge Objects 215
 - for movies 98
 - for tracking object's motion 155
 - for tracking object's position 155
 - for tracking performance 188
 - image, listed 90
 - in scripts 174
 - Internet-related 206
 - naming scheme for 44
 - numerical 167
 - overview of 157
 - providing feedback with 190
 - purpose of 158
 - sound, listed
 - step-by-step instructions for 165
 - system, using 165
 - text, listed 63

- troubleshooting 46
- types of 166
- video, listed 100
- where used 158
- video
 - combining sound with 93
 - controlling with functions 100
 - controlling with variables 100
 - creating 99
 - functions *See also* specific functions
 - functions, listed 101
 - integrating 99
 - requirements for 99
 - step-by-step instructions for 102
 - synchronizing with audio and graphics 93
 - using variables with 100
 - variables *See also* specific variables
 - variables, listed 100
 - working with 99
- Video for Windows drivers, in distribution package 57
- Video for Windows movies
 - in distribution package 57
 - support for 96
 - See also* movies
- video icons
 - role of 28
- video overlay drivers, in distribution package 57
- videodisc
 - distribution and 51
 - drivers, in distribution package 57
- VideoDone variable 100
- VideoFrame variable 100
- VideoResponding variable 100
- View menu
 - for showing or hiding toolbar 27
- voice-overs 94
 - See also* sound

W

- wait icons
 - role of 28
- web server, configuring for web-packaged files 208
- web-packaged pieces
 - step-by-step instructions for creating 55
- websites
 - See also* Internet
- WinCtrls.u32 231
- Windows applications
 - runtime applications 53
 - text fonts and 61

- Windows Control Get Property Knowledge Object 40
- Windows Control Knowledge Object 40
 - creating a control 233
- Windows Control Knowledge Object, using variables
 - in 245
- Windows Control Knowledge Objects 232
 - Get Property Knowledge Object 233
 - Set Property Knowledge Object 232
 - Windows Control Knowledge Object 232
- Windows Control Set Property Knowledge Object 40
- Windows controls and properties 243
- wizards
 - controlling targets with 221
 - description of 214
 - Knowledge Objects and 14
 - launching from Knowledge Objects 215
 - quitting 215
 - wizard user interface 220
- Word, RTF text in 62
- WordClicked variable
 - controlling text with 64
 - for tracking performance 189
- WordCount function
 - controlling text with 65
- WordCount variable 64, 127
 - controlling text with 64
- workspace, in icon-based authoring 23
- World Wide Web. *See* Internet
- Write INI Value Knowledge Object 40
- WriteExitFile function, sample use for 61
- WriteExtFile function
 - for saving user responses 192
- wrong responses, indicating 187
- WrongChoicesMatched variable 190

X

- XML parsed files 183
- XML Parser Xtra 183
- Xtra extensions
 - in distribution package 56, 57
 - role of 59, 86
 - scripting Xtra extensions 181
 - sprite Xtra extensions 178
 - transition Xtra extensions 152
 - types of 177
 - using 177

Y

- Yamaha MidXtra, child object created from 181