Everest Authoring System 2.0
Technical Reference

This on-line help system contains information about all object attributes, as well as A-pex3 programming commands. The Instructions item below contains brief help for procedural (how-to) topics, but for more details, refer to the Everest Tutorial or Design Guide books.

WHAT'S NEW (since initial 1.6 release)

WEB SITE

Visit our Internet Web site at `http://www.insystem.com` for the latest information about Everest.

VISUAL FEATURES

AnimPath type animation has been greatly enhanced for Picture objects by the addition of AnimCelStart, AnimCelEnd, AnimCelRate and AnimSpritePace attributes. These new features allow multi-frame, overlayed animation cels; plus, animation can now run in the processing background, so that other events can be processed simultaneously.

The BgndPicture, PictureFile and SPictureFile attributes now also support images stored in the following formats: .GIF, .JPG, .PCX, .RLE, .TGA, and .TIF. Also available (considered undocumented) are: .CUT, .GEM, .IFF, .MAC, .RAS and .WPG formats.

EraseType has been enhanced to give you better control of what is erased (including the AutoRedraw and BgndPicture images).

DrawText setting -2 solves disappearing text (it compensates for a Windows bug by enabling AutoRedraw).

The distance of the drop shadow drawn via DrawShadow can be influenced by setting Sysvar(199) to the desired number of pixels.

Everest offers an auto-ZIP check feature. At run time, if it cannot find a file , it will automatically search for one of the same name, except with a .ZIP extension. If found, it automatically unzips that file (to the temporary directory), then attempts to load the uncompressed file from there. This is ideal for

Inter/intranet stored projects, since .ZIPs are typically smaller, and can be downloaded faster. A new Project Packager feature can create the .ZIP files for you automatically.

Listbox objects can now be divided into multiple columns, thereby creating a grid. Refer to the ColChar, ColCount and Divider attributes. Comma delimited items in a text file can be loaded very easily into a multi-column Listbox with help from Fyl() function operation 9.

The AnimFile (Animate), FileName (Media) and SoundFile (Animate) attributes now support embedded files.

The Media object's ShowButtons attribute now lets you control the visibility of individual buttons.

INTER/INTRANET FEATURES

Everest now supports real-time playback and editing of projects stored anywhere on the Internet or intranets. To start playback within AUTHOR, from the Run menu, choose Start At, click the Direct button, and enter the URL address, book name and page name. For example, to run the Everest demo from our Web site, enter: `http://www.insystem.com/evdemo/@start;@start`. In ERUN, specify the URL via the ProjectPath entry in the EVEREST.INI.

NOTE: To actually communicate with the Internet/intranet, Everest requires that the computer already be properly configured. This means a working modem or other connection, a copy of WINSOCK.DLL, etc. In general, if the computer can connect to the Internet/intranet via a Web browser such as Netscape Navigator, or Microsoft Internet Explorer, then it should also connect properly with Everest. Also note that the computer must be able to connect directly; Everest will *not* work via proprietary on-line services such as America Online or CompuServe.

Everest also offers the Internet Simulator. Use the simulator to test run your projects if you do not have Internet access, or simply want to observe their execution speed at different communication rates. You can choose from a variety of simulated modem speeds.

The attributes AnimFile, BgndPicture, FileName, Pic, PictureFile, and SPictureFile that load files now also support URL addresses. This lets you easily download a file from an Inter/intranet site and incorporate it in your project. For more information, refer to the new help topic about file locations, Appendix F.

Fyl() function operations 8, 9 and 10 can load text files from an Inter/intranet site. At your request, operation 10 will also strip HTML codes from the file (to make it more readable in objects such as Textboxes).

The Project Packager can make your project granular, that is, break it into smaller pieces that can be downloaded more rapidly on the Internet. It places each page into its own book. At run time, Everest will automatically search for these granular books, so you need not modify your project's branching. At your option, during the packaging process, AUTHOR can lock each page to prevent editing (such as by other authors on the Internet), or apply a password to it to limit access.

You can edit projects stored on the Internet *on the fly*, that is, while test running them. Everest will save your changes, then automatically upload them to the Internet. To enable this feature, Everest needs to know your FTP information (upload directory, password, etc.). You can configure this information on on an expanded Settings window.

Ext() function operations 115, 116, and 117 construct and return a seemingly random password that can be used to restrict end-user access to your project. This can be handy if you are distributing your project via the Internet and want to allow access to only authorized end users.

Ext() function operation 123 returns the file date and time of the currently executing Everest .EXE (helpful for determining user's ERUN version).

Ext() function operations 130 and 131 return the currently running DOS and Windows version, respectively.

Fyl() function operations 41 and 42 download an Inter/intranet file given its URL.

Fyl() function operation 43 uploads a file to the Inter/intranet via FTP.

Rec() function operations 12 and 13 assists with Internet-related user records issues.

Rec() function operation 14 checks the user access list.

You can monitor the status of the Internet connection via the Internet History window. From the Author windows Window menu, choose INet History. At run time, you can also make it visible via Ext() function operation 126.

BRANCHING FEATURES

The BRANCH command supports two new keywords: @next and @back. These branch to the next and previous page, respectively, as listed in the Book Editor.

The NextActivator2 and NextActivator3 attributes have been removed. For upward compatibility, Everest automatically appends those of older pages to NextActivator.

The JudgeActivator2 attribute has been removed. For upward compatibility, Everest automatically appends those of older pages to JudgeActivator.

PROGRAMMING FEATURES

The A-pex3 Assistant makes its debut. The Assistant walks you through writing lines of A-pex3 programming code. It's very helpful while you are learning the language. Invoke the Assistant from the Program Editor window or Attributes window by pressing F12.

The Dat() function can now help you determine the number of days between two dates.

Ext() function operation 114 returns the number of devices installed on the computer that are capable of playing .WAV audio. If your project plays .WAV files, check this function at the start of the project, and if it returns 0, display a polite warning to the user.

Ext() function operations 115, 116, and 117 construct and return a seemingly random password that can be used to restrict end-user access to your project. This can be handy if you are distributing your project via the Internet and want to allow access to only authorized end users.

Ext() function operation 124 returns the number of open windows in your project; use this if you need to know when the user is closing the last window of your project, such as in a CloseEvent.

Ext() function operation 125 computes the value of an expression specified as the second parameter; this allows calculation of expressions constructed at run time.

Fyl() function operation -7 compresses files into .ZIP files.

Fyl() function operation -8 uncompresses files from .ZIP files.

Fyl() function operation -12 makes subdirectories.

Fyl() function operation 10 supports Inter/intranet URLs for loading text and HTML files.

Fyl() function operations 41 and 42 download an Inter/intranet file given its URL.

Fyl() function operation 43 uploads a file to the Inter/intranet via FTP.

The new Mid() function copies portions of strings.

The A-pex3 Program editor window has an instant evaluator feature. Want to instantly know the value of a variable from the last test run? Click on it, and choose Evaluator (or press Shift+F9).

The Variables window offers a find feature that searches the contents of all variables for the item you specify.

EDITING FEATURES

The VisualPage (formerly VisualScreen) editor can be scrolled if it is smaller than the VirtualWidth and/or VirtualHeight settings. To toggle scrollability, press Ctrl+Alt+S. Note: this is considered an experimental feature; due to a Windows bug, it does not operate correctly with Shape and Line objects.

The IconScript has been upgraded into the Book Editor. To open a book, double click on its icon in the Book Editor. To open a page, double click on its icon in the Book Editor.

The New Book button in the Page Selection Window now also lets you create a new subdirectory to hold the new book.

Books can be assigned passwords. For a book that has a password, AUTHOR will prompt you to enter the password before it will open the book for editing.

Everest's edit-on-the-fly feature also works for Inter/intranet projects. You can edit projects stored on the Internet *on the fly*, that is, while test running them. Everest will save your changes, then automatically upload them to the Internet. To enable this feature, you need to configure your FTP information on the newly expanded Settings window.

The new Embedded File Manager (on the Utilities menu) helps you maintain files embedded in the book. It includes a "Freshen All" feature that automatically updates (reloads) all embedded files (in case you've modified the originals).

The Arrange feature of the Book Editor now has convenient row and column options.

Note that many editing keypresses have changed.

Many windows have an Instructions menu item or button. Choose the item to view context sensitive procedural help for that window.

OTHER FEATURES

At your option, the Project Packager utility can automatically compress your project's files into .ZIP files, either individually or as a group. It can even compress the proper combination of files together to fit on your distribution diskettes most efficiently.

The new Zip Maker utility helps you convert files into .ZIP format. You can highlight a group of files, and tell Everest to create a .ZIP file for each, or to store them together in one .ZIP file.

The Project Packager can also generate an InstallShield-compatible SETUP script for your distribution disks. You'll need a copy of InstallShield Corporation's InstallShield program to compile the script. This new feature combined with the one described immediately above makes it a snap to prepare your Everest-created project for distribution.

The ERUN player program compresses the .SPW and .SP0 through .SP9 files that are part of a user's bookmark; the resulting file has an .EPW extension. This can result in a significant reduction of the disk space occupied by user records.

The IncludeScreen attribute has been renamed IncludePage.

Page names may now consist of up to 16 alphanumeric chararacters (previous limit was 8). For granular deployment over the Internet, do not use greater than 8 characters in page names.

If you type a blank space within a page name, Everest converts it into an underscore.

The default object names now use underscores instead of periods.

Even though Special objects created with prior versions of Everest are still supported in this version, the Special object is considered obsolete; its icon has been removed from the ToolSet. Use a Program object in place of the Special. If youd like to convert a Special object into a Program, in the Attributes window, from the Options menu, choose Convert to Program.

It is no longer necessary to distribute COMPRESS.DLL, MHPC200.VBX and MHIV200.VBX with ERUN. Instead, GVBOX.VBX and GVJPEG.DLL should be distributed (GVJPEG.DLL can be omitted if your project does not load any .JPG image files). Also, COMPPLUS.DLL and DSSOCK.VBX should be distributed.

KNOWN MODIFICATIONS THAT MAY IMPACT YOUR PROJECTS

With each new version of Everest, we make every attempt to maintain 100% upward compatibility with projects created with prior versions. Sometimes we accidentally modify something in Everest that changes how an old project runs; most of these modifications are caught by our diligent beta testers, and

are corrected. Very rarely, forces beyond our control cause us to make such modifications. Such changes are listed below.

Versions 1.5 and 1.6 did not handle Textbox AnimPath operation identically when DrawText was set to 0. Version 2 restores such animation to the way it operated in version 1.5. To maintain version 1.6 style operation, you will need to set DrawText to a non-zero value.

The SPicture object is now driven by a different .VBX (GVBOX.VBX instead of MHPC200.VBX). It is possible there may be some small variances in the appearance of images displayed in SPictures.

The Media object no longer attempts to open the multimedia device at run time if the Command attribute is empty.

Removing message -004 from the EVEREST.MSG file no longer disables user log on. Now, to do so, put `Logon=0` anywhere in the Settings section of the EVEREST.INI.

Abs() Function

Applies to A-pex3 programming

Description Returns the absolute value of a number.

Syntax abs(Numeric)

Details Pass the number as the Numeric parameter. For Numerics greater than or equal to 0, Abs() returns Numeric. For Numerics less than 0, Abs() discards the minus sign of the Numeric and returns the result.

Example The following example calculates the horizontal distance between the current mouse location and the right edge of the Picture with IDNumber 1:

```
horzdist = abs(mse(1) - Picture(1).Right)
```

Action Attribute

Applies to OLE object

Description Determines the operation to perform with the OLE Object.

Settings

0	Create a new instance of an object. To use this Action, set the ServerType attribute to (embedded), and set the Class and Protocol attributes as well.
1	Create a linked object from the contents of the file specified via SourceDoc. To use this Action, set the ServerType to 0 (linked), and set the Class, Protocol and SourceDoc attributes as well.
4	Copy an object to the Windows Clipboard.
5	Paste data from the Clipboard to the OLE object. To use this Action, set the Protocol and ServerType attributes.
6	Updates the OLE object with current data from the server.
7	Activates an object for an operation, such as editing. To use this Action, set the Focus, ServerShow and Verb attributes.
8	Sends the string in the Execute attribute to the server. To use this Action, set Protocol to StdExecute.
9	Close the object and the connection with the server.
10	Delete the object.
11	Save a client object to a data file.
12	Load a client object from a data file.
13	Convert the current object to the type specified by the ServerType attribute.

AddItem Attribute

Applies to Combo, Listbox objects

Description Adds an item to the bottom of the list, or, if sorted, to the correct sorted location. Write-only and available at run time only.

Example The following example loads the contents of the C:\AUTOEXEC.BAT file into the Listbox with IDNumber 1:

```
filename = "C:\autoexec.bat"
filenum = 1                    $$ arbitrary number
ecode = fyl(1, filenum, filename)
IF ecode = -1 THEN      $$ -1 means no error
  DO
    aline = fyl(11, filenum)  $$ read file
    IF sysvar(1) # -1 THEN OUTLOOP
    Listbox(1).AddItem = aline
  LOOP
  IF sysvar(1) # 62 then ecode = sysvar(1)
ENDIF
dummyvar = fyl(0, filenum)    $$ close the file
IF ecode # -1 THEN
  dummyvar = mbx("Error " + ecode, 0)
ENDIF
```

Notes After adding/changing items at run time, we recommend that you do not change appearance attributes of the object (such as FontSize); doing so might reset the item list back to its original state. This is due to a limitation in the MicroHelp controls that drive these objects.

Also see Item, ItemList, LastAdded, RemoveItem, Sorted

AdjustResponse Attribute

Applies to Combo, Input, Mask objects

Description Specifies whether the user's response should be converted to lower-case and have spaces removed for the purposes of answer judging.

Settings
Yes remove spaces, convert to lower-case letters
No do not adjust response

Details A user can type spaces and upper-case letters as the response to a question. Such characters, if left unchanged, can make it difficult for you to obtain an exact match for answer judging purposes.

If you enable AdjustResponse, upon judging, Everest first removes spaces from the user's response and converts letters to lower-case...then it compares this adjusted response to your list of anticipated answers.

In most situations you should enable AdjustResponse. Rarely should you disable it. Examples of situations when you would disable AdjustReponse include: when spacing is critical for response correctness, and when case is critical for response correctness.

Notes The setting of AdjustResponse does not alter the value copied into the ResponseVar. If you later wish to adjust the value in the ResponseVar, use the Lwr() function and ^# operator. For example, if the name of the ResponseVar is rvar, use:

```
rvar = lwr(rvar ^# " ")
```

Also see Answers1, Lwr() Function, Operators, ResponseVar

Alignment Attribute

Applies to	Button, Check, Input, Option, Textbox objects	
Description	Controls the justification and location of text within an object.	
Settings	0	left justify
	1	right justify
	2	center
Also see	MultiLine, VerticalAlignment, WordWrap	

AllOtherAction Attribute

Applies to — Wait object

Description — Determines the action to perform when an event code does not match any Activators in a Wait object.

Double click — Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.

Details — Think of AllOtherAction as the "else" clause of a Wait object. Enter something for AllOtherAction only if you want to perform some action when an event code did not already trigger a different xxxAction in the Wait object.

Some authors employ AllOtherAction to detect when the user presses any key. To determine which key the user pressed, examine the value in Sysvar(12). Keypress codes can be found in Appendix A.

Example — Since AllOtherAction has room for just one line of A-pex3 programming, many authors use the JUMP command to jump to a JLabel positioned immediately before a Program object found later in the page. The contents of that Program object might resemble:

```
kee = sysvar(12): sysvar(12) = 0
IF kee = 1112 THEN      $$ Shift+F1 key
  BRANCH section1
ELSEIF kee = 1113 THEN        $$ Shift+F2 key
  BRANCH section2
ELSE                          $$ else go back to Wait
  JUMP @wait
ENDIF
```

Notes — AllOtherAction is unique in that it does not automatically remove event codes from the Windows event queue. If you want to remove the event code, set Sysvar(12) to 0 as part of your AllOtherAction processing.

Also see — EventVar

AllowSelection Attribute

Applies to	Textbox object
Description	Determines whether words in the Textbox are highlighted when the user drags the mouse across them.
Settings	Yes words become highlighted No words do not become highlighted
Details	This feature is typically used in a hypertext setting when you want to allow the user to mark one or more words of text, and then perform an operation (such as look up a definition). To retrieve the highlighted word(s), use the SelText attribute.
Also see	SelText

Animate Object

Description — The Animate object displays Autodesk Animator files (those with 8.3 file name extensions .FLC, .FLI and .AAS). Optionally, audio can be played while the animation is running.

Attributes

AnimFile
AnimStoppedEvent
BackColor
Bottom
ClickEvent
Comment
DblClickEvent
DragMode
Enabled
EndFrame
FadeIn
FadeOut
Height
Initially
Iterations
Left
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
Name
Play
Position
SaveAsObject
SetFocus
SoundDevice
SoundFile
Top
Update
Visible
Width

Details — Only one Animate object is allowed per window.

The standard Width of Animator files is 320 pixels; the standard height is 200 pixels. The Animate object cannot scale the image. The Animate object is always displayed on top of other objects.

Autodesk and other vendors market graphics editors that produce .FLC, .FLI and .AAS files. Contact them for information.

Also see — AnimPath, CopyPic, Media Object, SpecialEffect

AnimCelEnd Attribute

Applies to	Picture object	
Description	Specifies the cel of the PicBin object that contains the last frame of the AnimPath cel animation.	
Settings	0	do not animate via PicBin object
	1 to 32000	PicBin cel number
Double click	Opens icon dialog box (lets you visually choose a frame from the current PicBin).	
Details	After the AnimCelEnd frame is displayed, Everest restarts the animation with the AnimCelStart frame.	
Also see	AnimPath	

AnimCelRate Attribute

Applies to Picture object

Description Specifies how rapidly to change the image during AnimPath cel animation.

Settings

0	default
1 to 32000	the number of animation steps per frame
-1 to -32000	same as 1 to 32000, except display cels in reverse order

Details When Everest displays your animation, it moves the Picture object step by step along the AnimPath. AnimCelRate determines how many steps are made between each frame of the animation. So, for example, if you want the next frame of the animation to appear with each step, set AnimCelRate to 1. If you want two steps to occur before the next frame is displayed, set AnimCelRate to 2.

Larger values for AnimCelRate make the animation appear to run more slowly (because the image does not change as rapidly).

Notes When animating long AnimPath distances (greater than 3 pixels), Everest breaks the movement into a series of smaller steps. This might result in your animation frames changing more rapidly than desired. If you want to prevent Everest from using small steps, start the AnimPath setting with `s|` (be sure to use a lower-case letter s).

Also see AnimPath

AnimCelStart Attribute

Applies to	Picture object	
Description	Specifies the cel of the PicBin object that contains the first frame of the AnimPath cel animation.	
Settings	0	do not animate via PicBin object
	1 to 32000	PicBin cel number
Double click	Opens icon dialog box (lets you visually choose a frame from the current PicBin).	
Details	AnimCelStart (and AnimCelEnd) are the keys to multi-frame animation. They let you specify the frames of a Picbin object to display during AnimPath animation. See AnimPath for instructions on how to create multi-frame animation.	
Also see	AnimPath	

AnimFile Attribute

Applies to	Animate object
Description	Specifies the name of the disk file that contains the animation.
Double click	Opens file dialog box. Double click on the file you want.
Details	Enter the name of the disk file that contains the animation you want to display. Animation files must be stored in Autodesk Animator format. Such files usually have name extensions of .AAS, .FLC and .FLI. For help with file locations, refer to Appendix F.
Also see	Animate Object, Sysvar(16)

AnimPath Attribute

Applies to Button, FlexText, Picture, SPicture, Textbox objects

Description Describes a relative path along which Everest moves the object at run-time.

Settings A string of up to 1024 characters that describes the path (note that the Attributes window is limited to 250 characters).

Double click Lets you drag object to create the path. See details below.

Details The AnimPath attribute lets you designate a path along which Everest will move the object at run time. You can create AnimPath manually, or let Everest generate the path for you.

ASSISTED ANIMPATH CREATION

Everest can assist you by recording an AnimPath. This feature is particularly helpful for non-linear paths. Here are the steps:

1) Position the object at the desired starting location in the VisualPage editor.

2) Double click on AnimPath in the Attributes window. A window will appear with brief instructions. When ready, click OK.

3) Point to one of the object's eight sizing handles in the VisualPage editor, and press and hold the RIGHT side mouse button.

4) Drag the object along the desired animation path. For easiest operation, avoid dragging for more than approximately 7 seconds.

5) Release the mouse button when done. The object will snap back to its starting location.

The steps above create AnimPath for you. If AnimPath can be expressed in 250 or fewer characters (the limit for the Attributes window), Everest sets the AnimPath attribute in the Attributes window. If your AnimPath is longer, Everest informs you, and puts the command in the Clipboard. You can then paste the command into a Program object.

AnimPath has the form:

[delta X] [, delta Y][Repeat]|...|[delta X] [, delta Y][Repeat]

For example, AnimPath might resemble:

```
2,4|-1|0|,2|3,3R3
```

which, at run time, Everest interprets as a series of five movement steps:

2,4 move the object 2 pixels right and 4 pixels down
-1 move the object 1 pixel left

0 pause briefly (approximately .05 seconds)
,2 move the object 2 pixels down
3,3R3 move the object 3 pixels right, 3 down and repeat 3 more times

The separator character between each step is | (ASCII 124).

To play back the AnimPath, run a Preview of the page.

MANUAL ANIMPATH CREATION

You can type the AnimPath manually in the format described above. This is easiest for linear animations. Here are a few examples:

`10r20` moves right 10 pixels, repeats 20 times

`2,2r100` moves down and right 2 pixels, repeats 100 times

`-10r20` moves left 10 pixels, repeats 20 times

`0,-5r2` moves up 5 pixels, repeats twice

ANIMATING ACROSS A BACKGROUND

To make the background show through during animation, use the following technique:

1) Use a Picture object to hold the image to be animated.
2) Enable the Picture's CopyBgnd attribute.
3) Set the Picture's TpColor attribute to the desired transparent color.

The background that shows during the animation can be influenced by the setting of the Layout object's AutoRedraw attribute. Try different AutoRedraw settings to find the desired effect and/or correct overlay problems.

At run time, the animation might begin with a flash. To eliminate this flash, set Initially to 2.

MULTI-FRAME ANIMATION

To change the image during the animation (for example, if you want to animate a person walking), use the following technique:

1) Gather all the frames of the animation into one .BMP bitmap file. Do so by using the graphics editor of your choice, and tiling all the frames (into any number of rows and columns you prefer). All the frames must be the same height and width.
2) Add a PicBin object to your Everest page, and set its BMPFile attribute to the name of the file you created in the previous step.
3) Set the PicBin's Rows and Columns attributes to match the number of rows and columns you used when tiling the frames.
4) Add a Picture object to your Everest page, positioned somewhere after the PicBin.
5) Set the Picture object's AnimCelStart and AnimCelEnd attributes to the first and last

frames of the animation, respectively. The easiest way is to double click on the attribute names; this displays the PicBin contents (your frames).
6) Create the AnimPath as you normally would.
7) Run a Preview to view the result.
8) To make the background transparent, set TpColor.

BACKGROUND (SPRITE) ANIMATION

To make your AnimPath animation run in the background, so that Everest can process the remaining objects in your page as well as events, refer to AnimSpritePace.

IN-PLACE ANIMATION

To make a multi-frame animation simply change appearance without moving around the window, use AnimSpritePace, and set AnimPath to

```
0|a
```

RANDOM PATH ANIMATION

To make a multi-frame animation move around the window randomly, use AnimSpritePace, and set AnimPath to

```
x,y|z
```

CORRECTING ANIMATION APPEARANCE PROBLEMS

The appearance of animation is influenced by many factors, including the size of the moving object, the speed of the computer, the speed of the video display adapter, and the setting of the Layout object's AutoRedraw attribute. Several AnimPath switches allow you to fine tune the appearance of your animation. In general, these switches consist of a single letter in which case is significant. When using a switch, place it at the start of the AnimPath, followed immediately by a `|` character. If you use multiple switches, separate each with the `|` character.

`L` engage visual lock; use it to help cure animation flicker; Everest automatically engages this feature for transparent overlays that employ AnimSpritePace

`l` (lower-case letter L); disengage visual lock; can help animation run faster

`T` engage timer delays; try it if animation runs too fast

`t` disengage timer delays; can help animation run faster (for animations in which AnimSpritePace is set to 0)

`U` engage object update; if multi-frame animation gets stuck showing one frame, or does not appear at all, try using this

`u` (lower-case) disengage object update; if the background behind an overlayed animation appears to vibrate, try using this

`W` engage window update; if an animating object leaves trails, try using this

`w` (lower-case) disengage window update; if the background behind an overlayed animation appears to vibrate, try using this

Example The following AnimPath example uses the U and W switches to force a visual refresh of both the animating object and the window:

```
U|W|10r10
```

Notes Everest inserts brief real time pauses between each AnimPath step so the speed of play back movement of the object should be similar on all computers. If you want the animation to run at maximum speed (i.e. without artificial delays), start the AnimPath with the letter `t` (be sure to use a lower-case letter t), followed by a `|` character. For example, your AnimPath might resemble: `t|3R50`.

The operation of AnimPath for Textbox objects is influenced by the setting of the DrawText attribute.

Due to the way Windows refreshes the display, moving an object off an edge of the window, and then back into the window may cause all or a portion of the object to disappear until the end of the animation.

At run time, if the Layout object's LockUpdate attribute is enabled, Windows discards the AnimPath motion of an object. Do not enable LockUpdate when using AnimPath; alternatively, disable it prior to encountering the object with the AnimPath via an A-pex3 command similar to the following:

```
Window(0).LockUpdate = 0      $$ turn off LockUpdate
```

Also see Animate Object

AnimSpritePace Attribute

Applies to Picture object

Description Enables background AnimPath animation, and specifies the duration of time (in milliseconds) between consecutive frames.

Settings

0	do not use background animation
1 to 32000	number of milliseconds between frames (1000 = 1 second)

Details Authors typically use the AnimSpritePace feature to execute an animation in the background. Note: background here does not mean the visual background, instead it means the "event" background. Such animation is sometimes called "sprite animation" because while the animation is running, Everest continues to process other events, such as user mouse clicks, etc. Therefore, AnimSpritePath is handy when you want your project to display an animation, but still respond immediately to user input, or perform other tasks (such as display a second animation simultaneously).

Everest does all this with help from an internal timer: the timer waits for a short interval, then tells Everest to update the animation (display the next frame, etc.). Via AnimSpritePath, you specify the length of this interval. Express the AnimSpritePath setting in milliseconds; a good value to try first is 100 (which is 0.1 seconds). Experiment. Larger settings for AnimSpritePath produce longer intervals between frames, making your animation run more slowly.

MULTIPLE SPRITES

At run time, Everest starts a sprite animation and then continues to process the objects of your page. Therefore, if you want multiple, simultaneous sprites, simply put multiple Picture objects in your page that each employ AnimSpritePace. The AnimSpritePace setting for all such Picture objects must be identical.

The theoretical maximum number of simultaneous sprites per window is 99. However, we have found that a practical maximum (due to the memory limitations of Windows) is 8.

While multiple sprites are animating, Everest must do a substantial amount of work to handle the possibility that the sprites might overlap. Consequently, multi-sprite animation runs more slowly. If you know your sprites will not overlap, you can reduce Everest's workload and let your sprites animate faster. To do so, start your AnimPath with `v|` (be sure to use a lower-case letter v).

CONTINUOUS ANIMATION

To run the animation again after it finishes, put `|a` at the end of the AnimPath.

To make the animation run backwards after it finishes, put `|b` at the end of the AnimPath.

Such animation continues to run until you delete the Picture object, or set AnimPath to "" via programming.

GENERATING EVENTS DURING THE ANIMATION

If you want to coordinate other operations (such as sound playback) with your animation, you can do so by placing the desired event code or A-pex3 programming command right inside the AnimPath. Simply prefix the event or command with `|e`. For example, an AnimPath that GOSUBs to the Program object named "cue" could resemble:

```
2,4|eGOSUB cue|3,3R3
```

Use this feature at your own risk because some operations (such as changing the size of the Picture objects) may cause the animation to fail.

Notes

Due to the design of PC computer hardware, the internal timer cannot wait less than 50 milliseconds. Therefore, if you set AnimSpritePace to a value less than 50, Everest attempts to compensate by displaying more than one frame per timer interval.

Also see

AnimPath

AnimStoppedEvent Attribute

Applies to	Animate object
Description	Event code to generate, or programming to perform, when the animation sequence stops.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	When your animation reaches the end and stops, you may want to perform some other action (such as starting a different animation, or branching to another page). The AnimStoppedEvent attribute lets you specify the event code to generate when the animation stops. A Wait object in the page can then detect and process the event, taking whatever action you want.
Example	To replay the animation after it finishes all Iterations, set the AnimStoppedEvent to: `Animate(1).Play = 1`
Also see	Animate Object, Wait Object

Answers1, Answers2 (, Answers3, Answers4, Answers5, Answers6, Answers7, Answers8) Attributes

Applies to Button, Check, Combo, HScroll, Input, Listbox, Mask, Option, VScroll objects

Description Specifies one or more anticipated correct responses.

Details Enter the strings or numeric values you want to compare to the user's response. When Everest encounters a Judge Object in the page, it compares the user's response with the Answers1, Answers2, ... ,Answers8 lines until a match is found. Everest stores the number of the Answers line (1 to 8) that contains the first match into the Judgment attribute for the object. If you specify a JudgeVar variable, Everest also stores the same number into it.

To provide feedback after judging, examine (such as via A-pex3 programming) the Judgment attribute, or Everest's automatic scoring system variables (Sysvar(5), Sysvar(6), Sysvar(105), Sysvar(106), Sysvar(175) and Sysvar(176)). Note that that automatic scoring system variables do not operate unless you assign a value to the Tries attribute. For further information, see the Design Guide or the examples in TEMPLATE.ESL.

Some objects (such as Input) returns the user's response as text; other objects return numeric values. The object class determines whether you should express the anticipated answers in Answers1 through Answers8 as text or numbers. Here is a list:

Object	Returns
Button	0 (up) or -1 (down)
Check	0 (unchecked), 1 (checked) or 2 (grayed)
Combo	the text in the top box
HScroll	a number between Min and Max
Input	the text in the object
Listbox	the string returned by TaggedList
Mask	the text in the object
Option	0 (empty) or -1 (selected)
VScroll	a number between Min and Max

So, for example, to judge a Button click as correct, set Answers1 for that Button to -1. Another example: for an Input object, if the correct answer is "cat" you would set Answers1 for that Input to the word cat.

For grouping purposes, you can enter more than one answer in each Answers attribute. Separate answers with the | character (ASCII 124). For example:

Answers1	blue|purple|azure
Answers2	red|orange

In this example, if the user's response is blue, purple or azure, a value of 1 is stored in the Judgment attribute. If the user's response is red or orange, a value of 2 is stored in the Judgment attribute. Many authors use an IF...THEN block in a subsequent Program object to test the value of Judgment and provide feedback.

To let the user try again, jump back to the Wait object, use another Wait object in the

page, or let Everest run the page to the end, in which case it searches for the Wait object nearest the end of the page and automatically jumps back to it.

An easy way to determine if any active question objects remain from the most recent Judge is to examine Sysvar(4). When a Judge object is processed, Everest counts the number of question objects that are skipped (because of a match with the Ignore answer list), or which were answered incorrectly and have additional Tries remaining, and stores the total in Sysvar(4). Fields that have no limit on number of Tries are not counted.

Special answer judging features can be employed by prefixing the anticipated answer with a certain code. These are described in the examples below.

Examples

SPELLING ALLOWANCE

To allow for variances in spelling, include wild cards in the anticipated response.

? matches one character
* matches zero or more characters

For example:

```
blu?|purple*|a?ure*
```

matches any of the following: blue, bluw, purple, purple heart, asure, azure is the color. Does not match: blue sky, perple haze, aure.

WORD SEARCH

Prefix the anticipated answer with =W= to search for key words in the user's response. For example:

```
=W=blue|=W=purple=W=azure
```

matches the following: blue, blue sky, purple and azure. Does not match: blu, purple, azure.

Note that omitting the | character between =W=purple and =W=azure tells Everest to check that both words are found in the user's response.

SOUND ALIKE

Prefix the anticipated answer with =S= to perform a phonetic sound-alike comparison (Everest employs a SOUNDEX algorithm). For example:

```
=S=blue
```

matches the following: blue, bloo, blu, blew. Does not match: brew, broom, burn.

PATTERN MATCH

Prefix the anticipated answer with =P= to perform pattern matching. Use the following symbols:

?	matches one character
*	matches zero or more characters
#	matches any single digit
[list]	matches any single character in list
[!list]	matches any single character not in list

List can be a series of characters, or a range. To employ a range, use a hyphen, and list the range in ascending order. For example:

```
=P=[a-zA-Z]#
```

matches any alpha character followed by one digit.

EQUIVALENT MATCH

If you do not include one of the prefixes (=W=, =S=, etc.), by default, Everest performs what it calls an "equivalent match." While you certainly can prefix an anticipated answer with =E= for an equivalent match, the =E= is redundant in this situation.

NOT A MATCH

To reverse the logic of comparison, use # in place of = in the prefixes. For example:

```
#E#blue
```

matches any user's response that is NOT blue.

NUMERIC RANGES

Everest normally performs string comparisons for answer judging. You may want to judge numeric values or ranges, particularly for HScroll and VScroll objects. Prefix the answer with any of the following symbols:

>	greater than value
<	less than value
=	equals value (numeric comparison)
#	does not equal value

For example:

`>50`	matches any response with a numeric value greater than 50.
`>50<60`	matches any response with a numeric value between 50 and 60 (non-

inclusive).

VARIABLES

To compare the user's response with the contents of variables and/or expressions, surround with { }. For example:

```
{varname}|=W={search}|>{value+1}
```

SPECIAL SYMBOLS

To match any character that Everest uses as a special symbol, surround the whole anticipated answer in quotes. For example:

```
"*.*"|"=W=something"
```

LITERAL INTERPRETATION

To match quotes in the answer, prefix the anticipated answer with the literal operator, =L=. For example:

=L="quoted match"

Also see AdjustResponse, AntIncorrect1, Ignore, JudgeVar, Operators, ResponseVar, Tries

AntIncorrect1 (, AntIncorrect2) Attributes

Applies to	Button, Check, Combo, HScroll, Input, Listbox, Mask, Option, VScroll objects
Description	Specifies one or more anticipated incorrect answers.
Details	Anticipated incorrect answers are often employed to match common incorrect user responses for the purposes of providing custom feedback: "That's a common response, but it is not correct. Here's why..." When the user's response matches an answer in AntIncorrect1, the Judgment and JudgeVar are set to -1. When the user's response matches an answer in AntIncorrect2, the Judgment and JudgeVar are set to -2. Everest uses negative numbers so that you can easily distinguish such matches from those for correct answers (which are indicated by positive numbers). For proper answer syntax, see the Answers1 attribute.
Also see	Answers1, Ignore, JudgeVar, ResponseVar

Arr() Function

Applies to A-pex3 programming

Description Returns the number of elements in an array.

Syntax arr("arrayname")

Details Pass the name of the array within quotes; do not include () characters inside the quotes.

If "arrayname" does not exist, -1 is returned.

If "arrayname" is a non-array variable, 0 is returned.

Example The following commands make sure the array named files contains at least 100 elements:

```
IF arr("files") < 100 THEN
  REDIM files(100)
ENDIF
```

Also see DIM, DELVAR, REDIM, Sysvar(0), Var() Function

ARROW Command

Applies to A-pex3 Xgraphics programming

Description Draws an arrow.

Syntax ARROW (X, Y, Length, Angle [, Color] [, HeadSize] [, HeadAngle])

Details The ARROW command draws a line of length Length to the X, Y location at angle Angle. By default, it draws two additional lines (the arrow head) at X, Y which have length Length/5 at a 45 degree angle with the main line.

Specify Length in units of pixels. If you specify a negative number for Length, Everest draws the arrow head at the other end of the line.

Specify Angle in degrees (0 to 360). Zero degrees produces an arrow that points to the right. Ninety degrees produces an arrow that points upward.

Include the Color parameter only if you want to draw the arrow with one of the 16 palette colors. If you omit Color, Everest uses the current foreground color.

Include HeadSize only if you want to employ somthing other than the default size (which is Length/5) for the arrow head lines. Specify in units of pixels.

Include HeadAngle only if you want to employ something other than the default angle (which is 45) for the arrow head lines. Specify in degrees.

Example The following example draws an arrow at a 45 degree angle on the Picture object with IDNumber 1:

```
sysvar(108) = 1          $$ set Picture IDNumber
STYLE (1, 6)       $$ set line thickness
ARROW (100, 100, 50, 45)
sysvar(108) = 0          $$ restore to normal
```

Notes For proper operation, be sure to include a space between ARROW and (.

Asc() Function

Applies to A-pex3 programming

Description Returns the numeric ASCII Code of the first character of String.

Syntax asc(String)

Details Pass the function a character string in String. Returns a value between -1 and 255, inclusive. The value -1 is returned if the character is a null string.

Examples The following example stores the number 69 in the variable named char because 69 is the ASCII code of the upper-case letter E:

```
char = asc("Everest")
```

To determine the ASCII code of a character located elsewhere in the string, make use of the ^^ operator. For example:

```
char6 = asc("Everest" ^^ 6)
```

Also see ASCII Code Table, Chr() Function, Cvi() Function

Assistant Window

The Assistant Window is accessible via the pull-down menus in both the A-pex3 Program Editor and the Attributes Editor. The Assistant Window helps you write individual lines of programming in Everest's A-pex3 language. The Assistant can be especially useful while you are learning the language. Experienced authors often bypass the Assistant and program directly with more speed.

Please note that the Assistant has been designed to handle the common programming tasks found in Everest projects, and therefore does not support *every* feature of the A-pex3 language.

General Information

In general, start at the top by choosing what you want to do. Additional frames of information will appear below based on your selections. Be sure to examine the drop down lists, because often the Assistant will be able to suggest likely choices for you.

In the "So far" box near the bottom, you can watch as the Assistant assembles your line of programming. The Assistant enables the OK button only if the syntax of the line of programming is acceptable. It is important to note that even if the syntax is correct, it does not necessarily mean your program will function as you intend. You should always test run your pages to check for execution or logic errors.

Functions

Only the simpler (single-parameter) A-pex3 functions are listed.

Atn() Function

Applies to	A-pex3 programming
Description	Returns the trigonometric arctangent of an angle.
Syntax	atn(Angle)
Details	Express Angle in radians. To convert from degrees to radians, multiply by (pi/180).
Example	The following example stores the arctangent of a 45 degree angle in the variable named myarc:

```
myarc = atn(45 * 3.141593 / 180)
```

Also see Cos() Function, Sin() Function, Tan() Function

Attributes

Applies to All objects

Description Every object in Everest has attributes that describe the qualities of that object.

Details Many object attributes are listed in the Attributes window. To open the Attributes window, double click on a line in the Book Editor window.

For example, the attribute named Top describes the vertical visible location of the object within the window. You can set the value of Top during design time by modifying its value in the Attributes window, or at run time via an A-pex3 program.

Unless otherwise noted in this documentation, an attribute can be modified at both design time and run time. Some can be modified or accessed only at design time, and certain others only at run time. A few are "read-only" (which means they cannot be changed) or "write-only" (which means they have no value to be examined).

The following is a list of all attributes, grouped by functionality. Some apply to more than one object. Refer to the individual description of each for details.

APPEARANCE

Alignment
AnimCelEnd
AnimCelRate
AnimCelStart
AnimPath
AnimSpritePace
AutoCenter
AutoRedraw
AutoResize
AutoScale
AutoSize
BackColor
BorderColor
BorderStyle
BorderType
BorderWidth
Bottom
BoxAlignment
BoxSize
Caption
CaptionColor
CaptionRotation
    ColChar
    ColCount
ControlBox
CopyBgnd
DisplayIn
Divider

DoEvents
DrawMode
DrawPause
DrawShadow
DrawText
EdgeDistInside
EdgeSize
EdgeSizeInner
EdgeStyle
EdgeStyleInner
EdgeStyleInside
EndFrame
FadeIn
FadeOut
FillBarColor
FillColor
FillStyle
FillValue
FocusRect
Font3d
FontBold
FontItalic
FontSize
FontStrikeThru
FontUnderline
ForeColor
GaugeStyle
HeadingSize
Height
HoldDown
HValue
Icon
Indent
Initially
InnerBottom
InnerLeft
InnerRight
InnerTop
ItemAlignment
ItemColor
ItemList
Iterations
JumpPointer
Left
LetterRotation
LightColor
LockUpdate
MaxButton
MaxDrop
MinButton
MousePointer

Move
MultiLine
NormalPointer
Orientation
OutlineStyle
Pic
PicChecked
PicDown
PicGrayed
PicPressed
PicUnchecked
PicUp
PopupPointer
Refresh
Relocate
Right
Scrollable
ScrollBars
ShadowColor
Shape
ShapePointer
ShowButtons
Sorted
SpecialEffect
Style
Tagged
TaggedList
Text
Tile
TitleBar
Top
TopIndex
TpColor
VerticalAlignment
Visible
VValue
Wallpaper
Width
WindowBorder
WindowLayer
WindowState
WindowStyle
WordWrap
X1
X2
Y1
Y2
Zev
ZOrder

DISK & DEVICE ACCESS

AnimFile
BgndPicture
BMPFile
CMIData
DeviceType
FileName
FontName
HyperFile
HyperTopic
IncludePage
SoundDevice
SoundFile
SourceDoc
SPictureFile

EVENTS

AnimStoppedEvent
BackActivator
ChangeEvent
ClickEvent
CloseEvent
CommentActivator
DblClickEvent
DoneEvent
DragDropEvent
EOFEvent
EventVar
InvalidEvent
GotFocusEvent
JudgeActivator
LostFocusEvent
MenuActivator
MouseLeaveEvent
MouseOverEvent
MouseStayEvent
MoveEvent
NextActivator
Other1Activator to Other8Activator
QuitActivator
ResizeEvent
TimeEvent
UpdateEvent

INTERACTION

AllowSelection
Answers1 to Answers8
AntIncorrect1 to AntIncorrect2

CMIData
DragMode
EOFContinue
EOFEvent
Format
GroupChoice
Grouped
Ignore
InputTemplate
InvalidEvent
JudgeVar
Judgment
LargeStep
Max
MaxLength
Min
PassChar
PopupMenu
Preset
PromptChar
ResponseVar
SelLength
SelStart
SelText
Step
TabOrder
TabStop
TagStyle
TextLength
Tries
Value

MISCELLANEOUS

BackUpStack
Class
Columns
CopyPic
EndAt
Focus
Format
hDC
hWnd
Item
ItemCount
ItemIndex
LookAt
MenuStack
NewMenu
Pause
Period

Protocol
Rows
ServerClass
ServerShow
ServerType
Silent
SourceItem
Special1 to Special8
StartAt
SystemModal
TaggedCount
TimeFormat
UpdateInterval
VirtualHeight
VirtualWidth
Wait

OBJECTS

Comment
Condition
Create
Destroy
DisableObjs
Enabled
EraseFromID
EraseToID
EraseType
Group
IDNumber
Name
SaveAsObject
Update

OPERATIONS

Action
AddItem
AdjustResponse
AllOtherAction
BackAction
Command
CommentAction
Create
Destroy
HelpAction
HyperAction
MenuAction
NextAction
Other1Action to Other8Action
QuitAction

RemoveItem
SetFocus
Verb

Examples

To refer to an attribute in an A-pex3 program, use a calculation similar to:

```
texttop = Textbox(1).Top
```

To refer to an attribute of an object located within another open window, use a calculation such as:

```
response = Window(2)!Input(1).Text
```

When running an A-pex3 program, Everest remembers the last object referenced. If you omit the object's name, Everest assumes you are referring to the most recent object in the same program:

```
w = Textbox(1).Width
h = .Height       $$ faster than Textbox(1).Height
```

To set an attribute of a group of objects with consecutive IDNumbers, use the following syntax (the word "to" must be in lower-case):

```
Shape(1 to 5).Visible = 0
```

Also see

Commands, Functions, Objects, Operators

Attributes Editor

The Attributes Editor, if closed or obscured, can be made visible via any of the following methods: 1) choosing it from the main Author window's Window pull-down menu, 2) pressing Ctrl+A while authoring, or 3) double clicking on an icon in the Book Editor. The Attributes Editor window lists in grid form the properties of the currently selected object. To select an object for editing click on it with the mouse in either the VisualPage editor or the Book Editor. Here are several common editing techniques:

CHANGING AN ATTRIBUTE

There are several ways to change the value of an attribute. First, click on the name of the attribute you wish to modify. The current value of the attribute is displayed above the grid. You can click on this value, then change it manually.

DOUBLE CLICKING

Everest provides editing assistance for many attributes. Those whose name ends with a question mark may be set to either Yes or No. You can double click on such an attribute to toggle its state.

Other attributes have a limited number of possible values. Such attributes typically have numeric values, and are displayed with a description within parentheses. Double clicking on these cycles through the range of possible values. To cycle in reverse order, hold down the Ctrl key while double clicking.

Still other attributes, such as BackColor or PictureFile, display a dialog box when you double click on them. The dialog box helps you select a value for the attribute.

BOLD vs. NON-BOLD

Certain attributes are displayed in a bold font. Such attributes are known as "local" or "instance" attributes. The difference is significant only when SaveAsObject is enabled. The SaveAsObject feature lets you maintain a master copy of an object, and refer to it repeatedly throughout the book. When SaveAsObject is enabled, the value of attributes is determined by the master copy of the object. However, the value of attributes displayed in bold is determined locally...that is, the value you set overrides that of the master object for such attributes.

When SaveAsObject is enabled, the background color of the Attributes window shifts as a reminder. If you change a non-bold attribute for such an object, the master copy is also changed. Make such changes carefully as they will have an effect everywhere the object is used.

INSTANCE OF...

To copy the attributes of another object (i.e. make it an instance of that object), from the Options menu, choose Instance of. Everest will display a list of objects in the book save previously with SaveAsObject enabled.

ATTRIBUTES ORDER

By default, the attributes are displayed in a functional order. If you prefer alphabetical, choose it on the Options pull-down menu.

ATTRIBUTE HELP

For context sensitive help for a particular attribute, click on it in the Attributes editor and press Shift+F1.
For a list of all attributes, see Attributes.

AutoCenter Attribute

Applies to	Layout object	
Description	Controls whether the window Top and/or Left attributes are adjusted by Everest to center the window on the display.	
Settings	0	do not center automatically
	1	center horizontally (automatically adjust Left attribute)
	2	center vertically (automatically adjust Top attribute)
	3	center both horizontally and vertically (automatically adjust both Left and Top attributes)
Details	The Relocate attribute must be enabled for AutoCenter to have any effect.	
Also see	Relocate	

AutoRedraw Attribute

Applies to Layout, Picture objects

Description Controls whether a separate copy of graphics (such as Xgraphics) subsequently drawn onto the window itself are saved in memory and refreshed automatically.

Settings
Yes refresh automatically
No do not refresh

Details When AutoRedraw is not enabled, subsequent graphics (such as those created via Xgraphics commands such as BOX and CIRCLE, and DrawText settings of -1) are drawn directly onto the window itself. If something obscures the window (for example, a Mbx() message box), and then is removed, the original graphics are *not* automatically restored.

However, when AutoRedraw is enabled, Everest saves a hidden copy of the image of the window, and therefore can and will automatically refresh the window from that copy whenever necessary.

The setting of AutoRedraw influences how graphics are erased from the window. See EraseType for more information.

Notes Enabling AutoRedraw has a significant drawback. The hidden image consumes memory in proportion to the size and color depth of the window. There is no way to know exactly how much memory will be consumed, so your project runs the risk of encountering an unexpected "Out of Memory" error. However, you can monitor available memory via the Fre() Function.

When AutoRedraw is enabled, Xgraphics drawn via A-pex3 programming do not appear within the window until Windows pauses to process its event queue. You can force the event queue to be processed (and the graphics to be displayed) by employing the Ext(101) Function.

As an alternative to AutoRedraw, consider the Refresh attribute of the Program object. Refresh can re-draw your Xgraphics by automatically re-executing Program objects when necessary.

Also see Ext(101) Function, Refresh

AutoResize Attribute

Applies to Layout object

Description Determines if Everest proportionally resizes all objects contained in a window when the size of that window changes. When enabled, Everest also automatically resizes new objects subsequently added to the window, and scales Xgraphics.

Settings
Yes resize all objects
No do not resize all objects

Details When AutoResize is No, and the user changes the size of the window, all objects stay at their locations within the window. For example, if the window is maximized, this typically results in empty space to the right and below the objects.

When AutoResize is Yes, Everest proportionally resizes the objects to match the new size of the window. Specifically, Everest adjusts the Left, Top, Width and Height attributes. Other attributes, such as FontSize, are not changed.

Notes For special applications, at run time, you can retrieve the horizontal and vertical scaling factors via the Ext(111) and Ext(112) functions, respectively.

Also see SCALE, Scrollable

AutoScale Attribute

Applies to	Media object
Description	Determines if Everest tells the Windows Media Control Interface to resize a multimedia element to fit within its DisplayIn container.
Settings	Yes scale the image No do not scale the image
Details	Windows can scale the visual components of many multimedia elements (such as Microsoft Video for Windows .AVI files). When you set AutoScale to Yes, Everest requests that Windows scale the image to match the size of the DisplayIn container.
Also see	DisplayIn

AutoSize Attribute

Applies to	Picture object
Description	Determines if the picture's box area is changed to match that of the image.
Settings	Yes adjust box size No do not adjust box size
Details	Note that enabling this feature does NOT scale the image to fit the box, but instead adjusts the size of the box to that of the image. If you need to scale a bitmapped picture, use the SPicture object instead of the Picture object.
Also see	Picture Object,, SpecialEffect, SPicture Object

BackAction Attribute

Applies to — Wait object

Description — Specifies the action to perform when the BackActivator event is triggered.

Double click — First: sets BackAction to BRANCH @prev. Next: Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.

Details — When a Wait object sees that an event code matches the BackActivator event, it traps that event code, and performs the BackAction.

Most authors employ the BackActivator and BackAction to branch back to a previous page. Everest maintains a list of previous pages in the backup stack system variables Sysvar(71) to Sysvar(78).

Examples — A common BackAction entry is

```
BRANCH @prev
```

which tells Everest to backup to the previous page viewed by the user.

Another common BackAction entry is

```
BRANCH @back
```

which tells Everest to backup to the previous page in the book.

Also see — BackActivator, BackUpStack, BRANCH, Wait Object

BackActivator Attribute

Applies to	Wait object
Description	Specifies the numeric event code that triggers the BackAction.
Settings	-32000 to 32000, or a string surrounded by quotes
Double click	Opens event code dialog box. Press the desired key to automatically generate the corresponding event code.
Details	Everest watches the events that occur in your project, and checks if one matches the event code you specify as the BackActivator. If a match is found, the event is removed from the queue, and Everest performs the BackAction. Most authors employ the BackActivator to detect when a user has pressed the "back up to previous page" key.
Example	To make a Ctrl+B keypress the event that invokes the BackAction, set the BackActivator to the event code for Ctrl+B: 2066.
Also see	BackAction, Wait Object

BackColor Attribute

Applies to Animate, Flextext, Input, Layout, Mask, OLE, Picture, SPicture, Textbox objects

Description Sets the color that appears behind the text or image in the object

Double click Opens color dialog box. Click on the color of your choice.

Details To view and select colors from the standard Windows color dialog, click on the "more information" down arrow to the right of the attribute editing field. A color dialog box will appear; choose the desired color.

Experienced authors can type the numeric color code directly in the attribute editing field. The color values are displayed in hexadecimal format. All colors are made of a mixture of red, green and blue. The amount of each of three primary colors is specified in the hexadecimal color value as follows:

&Hbbggrr

where

&H	indicates hexadecimal format
bb	is the amount of blue (00 to FF)
gg	is the amount of green (00 to FF)
rr	is the amount of red (00 to FF)

Since each of the red, green and blue values can range from 0 to 255, there are 256 * 256 * 256, or 16.7 million, possible color combinations. On most typical computers today, Windows can only display 256 different colors at once. Windows automatically attempts to find a color in its palette that most closely matches the color you selected. It might do so by dithering the color (mixing two or more different colors); dithered colors do not work well as text backgrounds. You'll need to experiment to find acceptable colors for your application.

SYSTEM COLORS

You might wish to design your project to employ system colors (those colors that the user defines within Windows). To do so, enter one of the following special codes as the color attribute:

&H80000000	scroll bars gray area
&H80000001	desktop
&H80000002	active window caption
&H80000003	inactive window caption
&H80000004	menu background
&H80000005	window background
&H80000006	window frame
&H80000007	text in menus
&H80000008	text in windows
&H80000009	text in caption, size box
&H8000000A	active window border

&H8000000B	inactive window border
&H8000000C	background of multiple items
&H8000000D	highlight background
&H8000000E	highlight foreground
&H8000000F	face shading on buttons
&H80000010	edge shading on buttons
&H80000011	grayed (disabled) text
&H80000012	foreground of text on buttons

SETTING COLORS AT RUN TIME

You can change a color attribute at run time via A-pex3 programming. If you specify a color in hexdecimal form, be sure to surround it with quotes, or employ the Val() Function. For example:

```
Window(1).BackColor = "&H80000005"

Window(1).BackColor = val("&HFFFFFF")
```

Alternatively, you can employ the Rgb() Function. For example:

```
Window(1).BackColor = rgb(255, 255, 255)
```

Notes — There is a known bug in Windows that can cause the focus to move from one object to another when you choose a color from the color dialog window.

Also see — COLOR, FillColor, Rgb() Function

BackUpStack Attribute

Applies to Layout object

Description Controls whether the name of the page is appended to the backup stack system variable.

Settings
Yes append to backup stack
No do not append to backup stack

Details Everest can maintain a list of the pages the user has viewed. You can employ this list to easily back up to a prior page via the branching instruction:

```
BRANCH @prev
```

Typically, pages that refresh the entire window, such as those that start a new topic in your project, should be appended to the backup stack.

Pages that only update or modify objects placed in the window by prior pages are usually not appended to the backup stack. This is to avoid display problems if the user backs up from a subsequent page (one that no longer has the same objects within the window).

Also see BackAction, MenuStack

Bbt() Function

Applies to A-pex3 programming

Description Calls the Windows API BitBlt() (bit block transfer) function. Intended for use by experienced programmers.

Syntax bbt(hdcDest, XDest, YDest, Width, Height, hdcSrc, XSrc, YSrc, Rop)

Details The Bbt() function copies a bitmap from one object to another. It is often used to copy portions of the image in a Picture object to another (or same) Picture object. Returns a non-zero number if OK, 0 if an error.

The Rop parameter is a number that represents the raster operation to perform, as defined by Windows. A common Rop is hexadecimal value CC0020, which copies the image from the source to the destination. Consult a Windows Programmer's Reference for other Rop values.

Example The following example copies a rectangular area 50 pixels in width and height from the upper-left corner of the Picture object with IDNumber 1 to a location 100 pixels from the upper-left corner of the Picture object with IDNumber 2:

```
ok = bbt(picture(2).hDC,100,100,50,50,
picture(1).hDC,0,0,val("&HCC0020"))
```

Notes When the Source and Destination are the same object, if the image you are copying appears and then disappears immediately, you may need to reference function Ext(101) prior to using Bbt(). Doing so allows Windows to update the object properly.

Also see AutoRedraw, hDC

BgndPicture Attribute

Applies to	Layout object
Description	Specifies the name of the picture file to display as the background inside the window.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).

Settings

FileName	displays the specified picture file from disk
|FileName	displays the specified picture file from the book
(clear)	removes a prior image from the window
>0	displays PicBin image (cel number)
<0	copies image from the Picture object with the IDNumber specified

Details

Everest can display a .BMP, .GIF, .JPG, .PCX, .RLE, .TGA, .TIF or .WMF file as the background in a window. Enter the name of the file you want, or leave the attribute empty for no change to the background picture.

To remove a picture from the window, enter "(clear)" as the name of the file, or use EraseType settings 65 or 67.

To display an image from the PicBin object, enter a numeric value greater than 0.

To tile an image to create a background pattern, enable the Tile feature.

To scale a BgndPicture image at run time to match the size of window, employ a SpecialEffect.

BgndPicture images appear in a layer behind all other graphics, making them useful for backdrops or when you need to draw other graphics (such as labels or arrows) on top.

For help with file locations, refer to Appendix F.

Notes

If you wish to draw Xgraphics on top of a BgndPicture, for best results you should enable the Layout object's AutoRedraw attribute.

Also see

EraseType, GFILL

BMPFile Attribute

Applies to	PicBin object
Description	Specifies the name of the disk file that contains the picture to load into the picture bin.
Double click	Opens file dialog box. Double click on the file you want.
Details	The picture file must have been saved in .BMP format. All the individual images (the cels) that make up the image must have the same height and width. For help with file locations, refer to Appendix F.
Notes	Everest does not adjust to the palette contained within the BMPFile. If the colors of your BMPFile do not show up properly when you display the cels, then also load the same BMPFile into an invisible SPicture object located anywhere on the page.
Also see	Columns, PicBin Object, Rows

Book Editor

The Book Editor window, if closed or obscured, can be made visible by choosing it from the main Author window's Window pull-down menu, or by pressing Ctrl+B while authoring. The Book Editor window lists in flowchart/columnar form icons for books, pages and objects. Books contain pages, and pages contain objects. Objects are items such as Textboxes, Pictures and Buttons. In general, to modify one of these items, you first highlight it by clicking or double clicking the mouse on it. Here are several common editing techniques:

POINTER - General Information

The Pointer is the arrow that appears in the Book Editor to the left of the icons. The Pointer marks where the next (i.e. new) icon will be inserted. You can drag the Pointer with the mouse. Everest also automatically moves the Pointer when you drag icons within the Book Editor.

BOOKS - Opening

To open a book (and view/edit the pages within), double click on its icon. Book names are displayed using upper-case letters and a bold font, and resemble BOOK1.ESL. Only one book can be open at a time. The name of the currently open book is displayed as the caption for the Book Editor window, and the word "open" is displayed within the book editor.

BOOKS - Opening Different Subdirectory

The Book Editor shows all the books in the current subdirectory. To use a different subdirectory, from the Author window's File menu, choose New.

BOOKS - Creating New

To create a new book, from the Book menu, choose New. You'll be prompted to enter the name of the book; enter up to 8 alphanumeric characters. If you would like to also create a new subdirectory to hold the new book, simply prefix the book name with the subdirectory name. For example, if the window shows the current location to be C:\EVEREST, if you enter `project1\book1`, Everest will create C:\EVEREST\PROJECT1\BOOK1.ESL.

BOOKS - Closing

To close an open book, double click on its icon. Alternatively, open a different book.

BOOKS - Copying

To copy an entire book, close it first, then hold down the Shift key and drag the desired book's icon using the right side mouse button. Everest will then ask you to enter a name for the copy.

BOOKS - Deleting

To delete a book, highlight it and from the Edit menu choose Delete. Careful: deleting a book deletes all its pages!

BOOKS - Finding a Page

To find a book that contains a page with certain text, from the Book menu choose Find. This feature is helpful when you do not remember the name of a page, or in which book it is located.

BOOKS - Passwording

To assign an editing password to a book, first open the book, then from the Book menu choose Password. Once you assign a book a password, when you later open the book for editing, Everest will ask you to enter that password. Passwords are a convenient way to discourage others from editing the pages of your book.

PAGES - Opening

To open a page (and view/edit the objects within), first open the book in which it is located (if necessary), then double click on the page icon. Only one page can be open at a time. Page names are displayed using lower-case letters and a bold font. The word "open" is displayed next to a page that is currently open.

PAGES - Creating New

First, move the Pointer to the desired location for the new page, then from the Page menu choose New. Alternatively, drag a Page icon from the Toolset and drop it at the desired location in the book.

PAGES - Copying Within Same Book

To copy a page (for example, to use it as a template), hold down the Shift key and drag the desired page's icon using the right side mouse button. When you drop the icon, Everest will prompt you to enter a new name for the page; each page in a book must have a unique name.

PAGES - Copying Between Books

To copy a page from another book, first open the book that contains the page(s) to copy. Highlight the desired page(s) and from the Edit menu choose Copy. Then open the book where you want to place the copies. Move the pointer to the desired location, and from the Edit menu choose Paste. Alternatively, to copy a single page to another book in the same subdirectory, hold down the Shift key, drag the desired page's icon via the right-side mouse button, and drop it onto the destination book's icon. Everest will automatically open that book, and place the page at its end.

PAGES - Copying Objects into Current Page

To add the objects of one page to another, first open the destination page, then drag the source page's icon into it.

PAGES - Re-ordering

To move a page to a different location in the book, simply drag its icon with the right mouse button. Do not hold down the Shift key.

PAGES - Finding by Name

To find a page within the current book, first click on the Book Editor window (to make it the active window), then press the first letter of the page's name.

PAGES - Finding by Content

To find a page within the current book that contains certain text, from the Page menu, choose Find.

OBJECTS - Editing

To edit an object (such as a Textbox or Picture), first open the page that contains the object, then click on the desired object. Its attributes will appear in the Attributes window. For Program and Menu objects, double click on the icon to open the editing window. Object names appear in lower-case letters and a non-bold font.

OBJECTS - Creating

To create a new object, drag its icon from the Toolset and drop it on either the VisualPage editor or the Book Editor.

OBJECTS - Copying

There are several ways to copy an object. Within the Book Editor, hold down the Shift key and drag the source object's icon via the mouse (use the right side mouse button), then drop it at the desired location in the page. Alternatively, in the Book Editor, click on the object to copy (in order to highlight it), then hold down the Shift key while dragging an icon (of the same class) from the Toolset, and drop it on the VisualPage editor.

OBJECTS - Renaming

To change the name of an object, first highlight it, then from the Edit pull-down menu, choose Rename.

OBJECTS - Toggling

Objects can be "toggled" off and on. An object that is toggled off is ignored when the page is run. Authors often temporarily toggle off certain objects for debugging purposes. To toggle an object, highlight it, then from the Object menu choose Toggle. The Book Editor displays $$ next to the icons of objects that are toggled off. To toggle the object back on, repeat the process.

OBJECTS - Hiding

On very complex pages, objects may overlap and become difficult to edit. To temporarily hide an object (so that others beneath it can be edited more easily), highlight the object to hide, then from the Object menu choose Hide. The Book Editor displays H next to the icons of objects that are hidden. To make the object visible again, repeat the process. Note: Hide works only during editing; to alter the visibility of an object at run time, use the Initially attribute.

OBJECTS - Asterisk

An asterisk (*) near an object's icon indicates that you have changed that object. Everest removes the asterisks when you save the changes.

OBJECTS - Commenting

To write a comment about an object, first highlight the object, then double click on the comments box near the bottom of the Book Editor.

BorderColor Attribute

Applies to	Button, Check, Combo, Frame, Gauge, Line, Listbox, Mask, Option, Shape objects
Description	Controls the color of the edge of the object.
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.

BorderStyle Attribute

Applies to	Button, Check, Frame, Gauge, Input, Option, Textbox objects
Description	Sets the thickness of the outline edge of the object.
Settings	0 no border 1 thin border 2 thick border (Input and Textbox only)
Details	For Input and Textbox objects, the color of this border is preset by Windows; you cannot change it.
Note	Due to a bug in MicroHelp's Input and Textbox objects, when BorderStyle is 2 and scroll bars are displayed, the window's background might show within a portion of the object.
Also see	Indent, WindowBorder

BorderType Attribute

Applies to Listbox object

Description Sets the type of edge and caption displayed with the Listbox.

Settings

0	no border
1	thin
2	sizable
3	thin with caption
4	sizable with caption

Details To display a Caption at the top of the Listbox, you must use a BorderType value of 3 or 4.

Listboxes with BorderType 3 and 4 can be relocated within the window by the user at run time (dragged by the caption bar with the mouse).

Also see Caption

BorderWidth Attribute

Applies to	Line, Shape objects
Description	Sets the thickness of the edge of a shape or line.
Settings	1 to 8192, inclusive
Details	If you want to be able to control the appearance via the OutlineStyle attribute, you must set BorderWidth to 1.
Also see	OutlineStyle

Bottom Attribute

Applies to	Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects
Description	Controls the location of the bottom edge of the object. Available at run time only.
Settings	A value larger than that in the Top attribute.
Details	Specify in units of pixels. The value of Bottom is the same as Top + Height.
Example	The following A-pex3 command draws a line from the bottom-right corner of the Picture object with IDNumber 1 to the upper-left corner of the Picture object with IDNumber 2: LINE (Picture(1).Right, .Bottom, Picture(2).Left, .Top)
Also see	Height, Move, Top

BOX Command

Applies to A-pex3 Xgraphics programming

Description Draws a rectangle.

Syntax BOX (X1, Y1, X2, Y2 [, Color])

Details The BOX command draws a rectangle. Specify the coordinates of two opposite corners in pixels via the X1, Y1, X2 and Y2 parameters. Include the Color parameter only if you want the edge of the box to be drawn using one of the 16 palette colors; otherwise Everest uses the current foreground color.

The box is drawn using the current STYLE command settings. The STYLE settings control the appearance of the edge as well as the inside of the box (whether it is transparent or filled).

Example The following example draws an empty (transparent inside) bright green box:

```
STYLE (4, 1)
COLOR (-1, 0, 255, 0)
BOX (50, 50, 100, 100)
```

Notes For proper operation, include a space between BOX and (.

The box is filled according to the current FillStyle set by a previous STYLE command. To draw an empty box, first set FillStyle to 1 via a STYLE (4, 1) command.

Also see FBOX, RBOX, STYLE

BoxAlignment Attribute

Applies to	Check, Option objects	
Settings	0	left side
	1	right side
Description	Determines on which side of the object the box (for the check) or circle (for the option) is placed.	
Also see	BoxSize	

BoxSize Attribute

Applies to	Check, Option objects
Settings	0 to 32,767
Description	Controls the size of the box containing the check mark or circle containing the option mark.
Also see	BoxAlignment, PicChecked

BRANCH Command

Applies to A-pex3 programming

Description Causes execution to continue at another page via a "go to" type branch.

Syntax BRANCH PageName

Details BRANCH is the primary means of moving from one page to another. The BRANCH command is often used in the NextAction attribute, and typically resembles:

```
BRANCH page2
```

Execution of the current page ends upon a BRANCH (i.e. no further objects or programming commands in the page are executed). Furthermore, project execution does not automatically return to the page that contains the BRANCH command. Compare BRANCH with CALL (which executes a page as a subroutine) and OPEN (which does not terminate execution of the current page).

SPECIAL NAMES

PageName can also be certain special names:

@back	branch to the preceding page in the book (compare this with @prev; do not create a page named @back)
@comment	collect a user comment via Everest's built-in comment system (do not create a page named @comment)
@end	save bookmark/user records and exit project (do not create a page named @end)
@exit	exit project without saving bookmark/user records (do not create a page named @exit)
@finish	save bookmark/user records and branch to a page named @finish
@menu	branch to previous menu (last page added to the menu stack; do not create a page named @menu)
@next	branch to the next page in the book (do not create a page named @next)
@prev	branch to previous page (last page added to the backup stack, typically the one last viewed by the user; do not create a page named @prev)
@wait	stop executing the page and return to the closest Wait object above to wait for the next event (such as more input from the user); this can be used to exit a Program object; do not create a page named @wait

BRANCH TO DIFFERENT BOOK

To branch to a page located in another Everest book (.ESL file), prefix the page name with the book name and a semicolon. Do not include the .ESL file name extension. For example, the following command branches to the page named intro in the book LESSON2.ESL:

```
BRANCH lesson2;intro
```

If the new book is located in a different subdirectory (i.e. not that of the current book), prefix the book name with a disk path. For example:

```
BRANCH C:\math\lesson2;intro
```

FORCE BOOK RELOAD

To force Everest to reload a book from disk, prefix the page name with the book name and a comma. Do not include the .ESL file name extension. For example, this forces a reload of the NEWDISK.ESL book:

```
BRANCH newdisk,intro
```

The only situation in which you must force a book reload is when your project is running on diskettes (or removable media), spans diskettes, and you want the user to replace the diskette with another in order to continue.

BRANCH & DISPLAY IN DIFFERENT WINDOW

To display a page in a different window, suffix the desired window number (from 1 to 8) surrounded with [ ] after the page name. Examples:

```
BRANCH help[2]

BRANCH C:\math\lesson2;help[2]
```

When ready to remove a window started via this manner, use the Destroy attribute.

DRIVE LETTER SUBSTITUTES

You may not know the disk drive from which the user is running your project (i.e. perhaps they have installed it on drive D:, or E:, etc.). To branch to a book in a different subdirectory on the current drive, use ? in place of the drive letter. For example:

```
BRANCH ?:\physics\lesson1;intro
```

If you want to employ the DOS default drive, use @ in place of the drive letter. For example:

```
BRANCH @:\physics\lesson1;intro
```

If you want to employ the StarPath (designated in the EVEREST.INI file), use *: in place of the drive letter and path. For example:

```
BRANCH *:lesson1;intro
```

Notes

Except as indicated in this topic, when assigning names to pages you create, avoid using the @ character in the name to prevent conflicts with possible future special page names.

Do not use @next or @back branching if you plan to make the contents of your book granular for Inter/intranet delivery.

Also see

BackUpStack, CALL, Destroy, GOSUB, Include Object, JUMP, MenuStack, OPEN, SysVar Variables

Button Object

Description Use the Button object to supply "clickable" options to the user.

Attributes
Alignment
AnimPath
Answers1
Answers2
AntIncorrect1
BorderColor
BorderStyle
Bottom
Caption
CaptionColor
ClickEvent
CMIData
Comment
Condition
Create
Destroy
DragMode
EdgeSize
Enabled
FillColor
Font3d
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
GotFocusEvent
Group
GroupChoice
Height
HoldDown
IDNumber
Ignore
Initially
JudgeVar
Judgment
Left
LightColor
LostFocusEvent
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
MultiLine
Name
Pic

PicDown
PicPressed
PicUp
Preset
ResponseVar
Right
SaveAsObject
SetFocus
ShadowColor
TabOrder
TabStop
Top
Tries
Update
Value
Visible
WallPaper
Width
Zev
ZOrder

Details

Most authors use Buttons in two ways:

1) to allow users to respond to questions, in which case anticipated answers are entered with the Button.

2) to allow users to navigate or choose options, in which case the author assigns an event code to the ClickEvent attribute of the Button.

Be sure to place a Wait object in the page somewhere after the Buttons to allow the user a chance to respond.

Also see

ClickEvent, Wait Object

CALL Command

Applies to A-pex3 programming

Description Executes a page as a subroutine.

Syntax CALL <PageName>

Details Use CALL to run another project page and return when done. Note that CALL suspends execution of the current page until a RETURN command is encountered. Compare this to the OPEN command that does not suspend execution of the current page. Unless you must run the CALLed page as a subroutine, we recommend that you use OPEN instead of CALL.

Do not use CALL to execute a page that contains a Wait object. When a Wait object is executed, the user gains control over the window (and, for example, might close it manually). This can make it difficult to employ the RETURN command appropriately. If you are CALLing such a page for display in another window, we recommend using the OPEN command instead.

Each time you CALL a page, the name of the page with the CALL is inserted into the CALL stack in variable Sysvar(58).

To return from a CALL, use the RETURN command. To avoid consuming memory, your project must eventually RETURN from a CALL.

CALL AND OPEN WINDOW

To execute the CALLed page in a particular window, suffix the page name with the window number (1 to 8) inside [ ]. For example:

```
CALL help[2]
```

If the window is not already open, Everest creates it automatically.

CALL TO DIFFERENT BOOK

To call to a page located in another Everest book (.ESL file), prefix the page name with the book name and a semicolon. Do not include the .ESL file name extension. For example:

```
CALL lesson2;intro
```

If the new book is located in a different subdirectory (i.e. not that of the current book), prefix the book name with a disk path. For example:

```
CALL C:\math\lesson2;intro
```

FORCE BOOK RELOAD

To force Everest to reload a book from disk, prefix the page name with the book name

and a comma. Do not include the .ESL file name extension. For example:

```
CALL newdisk,intro
```

The only situation in which you must force a book reload is when your project is running on diskettes (or removable media), spans diskettes, and you want the user to replace the diskette with another in order to continue.

DRIVE LETTER SUBSTITUTES

You may not know the disk drive from which the user is running your project (i.e. perhaps they have installed it on drive D:, or E:, etc.). To call to a book in a different subdirectory on the current drive, use ? in place of the drive letter. For example:

```
CALL ?:\physics\lesson1;intro
```

If you want to employ the DOS default drive, use @ in place of the drive letter. For example:

```
CALL @:\physics\lesson1;intro
```

Notes

If you want to place additional A-pex3 commands on the same line after the CALL, separate them with a colon AND a space. For example:

```
CALL yourmom: calls = calls + 1
```

You can open a maximum of 8 windows at a time, though you will likely reach the resource limits of Windows before reaching those of Everest. Use the Fre() Function to monitor resources.

When using a CALL to execute a page within the current window, that page must not contain any Wait objects.

A Preview cannot open additional windows (such as via CALL).

Also see

BRANCH, GOSUB, Include Object, OPEN, RETURN

Caption Attribute

Applies to	Button, Check, Frame, Gauge, Layout, Listbox, Option objects
Description	Specifies the text displayed on the object.
Double click	Opens Character Table window. Click on a character to insert it into the Caption.
Details	To enable an access key for an object, include an ampersand (&) character immediately before the character in the Caption you want for the access key. The user can move the focus to the object/select the object by pressing Alt+ <the access key>. To display an ampersand in the Caption, use &&. For windows (the Layout object), the Caption appears only if the window has a TitleBar.
Notes	Due to a bug in the MicroHelp control that drives them, Frame objects might not display a Caption correctly if it contains digits separated by spaces. If answer judging of Buttons works properly when you click on a Button with the mouse, but not when you press the access key, try enabling HoldDown. If the access key character does not appear underlined in the object, try increasing the object's Height.
Also see	Text, TitleBar

CaptionColor Attribute

Applies to	Button, Check, Frame, Gauge, Option objects
Description	Sets the foreground color of the text caption.
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.

CaptionRotation Attribute

Applies to	Frame object
Description	Controls the angle at which the Caption is displayed in a frame.
Settings	0 to 360 (degrees)
Details	Use the CaptionRotation attribute to display text drawn at any angle. Note: only Windows vector fonts can be rotated ("Roman" is a vector font). Non-vector fonts will not rotate. The following vector fonts are supplied with Windows 3.1: Modern, Roman and Script.
Notes	Windows controls how the text is rotated, and appears to have some problems doing so. Some values of CaptionRotation may produce unusual angles. You will probably need to experiment. Use at your own risk.
Also see	FontName, LetterRotation

ChangeEvent Attribute

Applies to	HScroll, Input, VScroll objects
Description	Event code to generate, or programming to perform, when the value of the object (i.e. the Text of the Input object, or pointer on scroll objects) changes.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	When you enter a number or string constant for ChangeEvent, you are merely telling Everest what event to generate when the user clicks on the object. To make use of that event (i.e. detect it and do something useful), you must include a Wait object in your page.
Example	Some authors use the ChangeEvent to detect when the user has changed the location of the pointer on the scroll bar or contents of an Input object, and then update another object.
Notes	The ChangeEvent fires only when the object has the focus. Due to a bug in Windows, avoid using function Ext(101) while processing a ChangeEvent. Due to a limitation in Windows, avoid using the Ibx() and Mbx() functions while processing a ChangeEvent. Due to a bug in Windows, TabStop must be enabled for the object to receive the focus via a mouse click. Due to a bug in Windows, do not have the Debug window open while processing a ChangeEvent.

Check Object

Description Use several Check objects to provide users with a list of items from which they can select one or more.

Attributes
- Alignment
- Answers1
- Answers2
  - AntIncorrect1
- BorderColor
  - BorderStyle
- Bottom
- BoxAlignment
- BoxSize
- Caption
- CaptionColor
- ClickEvent
  - CMIData
- Comment
- Condition
- Create
- Destroy
- DragMode
- EdgeSize
- EdgeStyle
- Enabled
- FillColor
- Font3d
- FontBold
- FontItalic
- FontName
- FontSize
- FontStrikeThru
- FontUnderline
- GotFocusEvent
- Height
- IDNumber
- Ignore
- Initially
- JudgeVar
- Judgment
- Left
- LightColor
- LostFocusEvent
- MouseLeaveEvent
- MouseOverEvent
- MousePointer
- Move
- MultiLine
- Name
- Pic

PicChecked
PicGrayed
PicPressed
PicUnchecked
Preset
ResponseVar
Right
SaveAsObject
SetFocus
ShadowColor
State
TabOrder
TabStop
Top
Tries
Update
Value
Visible
WallPaper
Width
Zev
ZOrder

Also see Option Object

Chr() Function

Applies to	A-pex3 programming
Description	Returns the character represented by an ASCII Code.
Syntax	chr(Numeric)
Details	Numeric must range from 0 to 255.
Example	The following example stores @ in the variable named atchar because 64 is the ASCII code of the @ symbol: `atchar = chr(64)`
Also see	Asc() Function, ASCII Code Table, Mki() Function

CIRCLE Command

Applies to A-pex3 Xgraphics programming

Description Draws a circle, ellipse or arc.

Syntax CIRCLE (X, Y, Radius, [Color], [StartAngle], [EndAngle], [Aspect])

Details The CIRCLE command draws a circle, ellipse or arc. The X and Y parameters specify the center of the drawing. Radius is the size (in pixels) of the drawing.

Include the Color parameter only if you want to draw using one of the 16 palette colors. Otherwise, Everest uses the current foreground color (set via the COLOR command).

Use StartAngle and EndAngle to control the beginning and end of an arc; specify in degrees (from 0 to 359.9). To draw a complete ellipse, or a painted one, omit these two parameters.

Aspect controls the ratio of the vertical to horizontal size of the drawing. Values between 0 and 1 produce a wide drawing. Values above 1 produce a tall drawing. For a true circle, omit Aspect.

Examples The following example draws a white circle filled with red lines:

```
COLOR (-1, 255, 255, 255)      $$ edge color
COLOR (-2, 255, 0, 0)          $$ inside color
STYLE (4, 5)                   $$ fill with lines
CIRCLE (100, 100, 50)
```

The following example draws an elliptical arc in the current foreground color:

```
CIRCLE (100, 100, 50, , 45, 180, .5)
```

Notes For proper operation, include a space between CIRCLE and (.

A circle is filled according to the current FillStyle set by a previous STYLE command. To draw an empty circle, first set FillStyle to 1 via a STYLE (4, 1) command.

Also see STYLE

Class Attribute

Applies to	OLE object
Description	Determines the class name of an embedded object.
Details	This attribute determines the type of object that is placed in the OLE object when the Action attribute is set to 0 (Create New) or 1 (Create from File). Check the documentation for the server application to determine the available Classes. You can also click in the Attributes window for more information, and Everest will display a list of Class names available on your computer.
Notes	Class names are sometimes case sensitive; it depends on the server application.
Also see	Action

ClickEvent Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, Layout, Listbox, OLE, Option, Picture, Shape, SPicture objects

Description — Event code to generate, or programming to perform, when the user clicks a mouse button while pointing to the object.

Settings — -32000 to 32000, or a string surrounded by quotes
or
any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN

Double click — Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.

Details — Most authors use the ClickEvent to determine when the user has clicked the mouse on an object (such as a Button), in order to then do something, such as branch to the next page, play music, or whatever. See the examples section below.

Most of the time, you enter a number or string constant for ClickEvent. Doing so merely tells Everest what event to generate when the user clicks on the object. To make use of that event (i.e. detect it and do something useful, like BRANCH to another page), you must place a Wait object later in your page. In that Wait object, enter the same event in one of the xxxActivators (such as NextActivator) and the desired action in the corresponding xxxAction (such as NextAction). Event codes generated by keypresses are listed in Appendix A.

Alternatively, you can also enter A-pex3 programming directly in the ClickEvent. If the desired programming does not fit on one line, consider putting it into a Program object that you invoke via a GOSUB command.

CLICKEVENT FOR SHAPES

The ClickEvent for a Shape Object differs in the following ways:

1) The click area is always rectangular (even if the Shape is a circle, ellipse, etc.). Additionally, Shapes have the highest priority for clicks, that is, if a Shape is obscured by (hidden behind) another object, Everest processes the Shape's ClickEvent first. This makes Shape objects ideal for hidden "hot spots" on Pictures, etc.

2) If you do not enter a ClickEvent for a Shape, or if the Shape's Visible attribute is 0, Everest processes the user's click as if the Shape were not present at all. That is, the ClickEvent for an obscuring object (or for the window) will be fired. If you want the Shape to be invisible, but still trap clicks, set its OutlineStyle to 0 (transparent) and its FillStyle to 1 (transparent).

3) The Shape's ClickEvent fires when the user presses a mouse button. For other objects, the event fires when the user releases the mouse button.

Examples — For branching, most authors assign the ClickEvent attribute a value identical to that of a certain key's event code. For example, PgDn can be the "next page" key in your project.

From Appendix A, you learn that PgDn generates an event code of 34. Therefore, to make a Button work the same as the PgDn key, enter `34` for its ClickEvent attribute. Then, to branch to the next page when the user either presses PgDn or clicks on the Button, put a Wait object in the page, and set NextActivator to the same event code, `34`. Finally, set NextAction to `BRANCH @next`.

There are several ways to play music upon the ClickEvent...here's one. Put a Media object in your page, just above the Wait object. In the Media object, set the DeviceType attribute to "WaveAudio" and the FileName to whatever .WAV file you want. Be sure to leave the Command attribute empty. Then, in the ClickEvent for a Button (or whatever object), enter the following:

```
Media(1).Command = "prev": Media(1).Command = "play"
```

You'll need to run a preview of the page to test this and hear the music.

Notes

You can determine which mouse button(s) the user clicked by examining the value in Sysvar(11) as part of the ClickEvent handling. The status of shift keys is also revealed by Sysvar(11); see Appendix A.

If the user clicks on a Shape that is on top of another object, the ClickEvent for the Shape fires upon mouse button down. Upon mouse up, the ClickEvent for the object underneath may or may not fire, depending upon the internal workings of that object (as defined by Microsoft and other third parties).

Also see

DblClickEvent, ShapePointer, Wait Object

CloseEvent Attribute

Applies to Layout object

Description Event code to generate or programming to perform when the window closes.

Settings -32000 to 32000, or a string surrounded by quotes
or
any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN

Double click Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.

Details Most authors use the CloseEvent to detect if the user has manually closed a window (such as by choosing Close from the Window's Control box).

Before it triggers the CloseEvent, Everest places a numeric code in Sysvar(172) that describes why the window is closing. Here is a table of Sysvar(172) values:

0 The user is attempting to close the window manually.
1 You unloaded the window via programming (such as the Destroy attribute).
2 The current Windows environment session is ending.
3 The Windows Task Manager is closing the window.

Examples If the user is closing the window manually (i.e. is Sysvar(172) = 0), you can override the closure by immediately setting Sysvar(172) to -1. To do so, for CloseEvent you would enter:

```
IF sysvar(172) = 0 THEN sysvar(172) = -1
```

If you would like to branch to the @finish page if the user closes the last window manually, set CloseEvent to (all on one line):

```
IF ext(124) = 1 & sysvar(172) = 0 & sysvar(18) # "@finish" THEN
sysvar(172) = -1: BRANCH @finish
```

Also see ClickEvent, ControlBox, MoveEvent, ResizeEvent, Wait Object

CMIData Attribute

Applies to — Button, Check, Combo, HScroll, Input, Mask, Option, VScroll objects

Description — Determines whether Everest saves Computer-Managed Instruction (CMI) data about the user's response.

Settings

0	do not save response
1	save adjusted response & judgment
2	save exact response & judgment

Details — When user records (i.e. log on and log off) are employed, your project can also collect CMI data. In terms of an analogy, if user records are a photograph, CMI is a videotape. When CMIData is enabled, upon answer judgment, Everest writes a record into a file named CMIFILE.DAT that contains the user's response and the answer judgment rendered

Additionally, with each record, Everest writes unique user identification information, the name of the page, the date and time, an indication of what is being saved ("r" for adjusted response, "x" for exact), and the IDNumber of the question object. For disk space consumption estimates, figure an average of 80 bytes per record.

"Adjusted response" means a copy of the users response after Everest removes spaces (ASCII 32) from it, and converts upper-case letters to lower-case.

In the record, Everest does NOT save the class of the object. Therefore, it is important that if, on a given page, CMIData is enabled for question objects belonging to more than one class (say, Input and VScroll) that you use different IDNumbers for each object. Doing so will help you identify the object in your CMI data analysis.

Everest writes the information to the CMIFILE.DAT file located in the same subdirectory as the user records file. The CMIFILE.DAT can be processed and analyzed with the SUMCMI.EXE program.

Notes — To write other data into CMIFILE.DAT, use the Rec() Function.

When saving CMI data about Button and/or Option objects used as a group (i.e. where the user makes a choice of one of several), usually you should enable the Judge object's Grouped attribute. For Grouped objects, CMIData saves the Group number, the judgment rendered, and the IDNumber of the object the user chose.

Also see — AdjustResponse, Rec() Function

ColChar Attribute

Applies to Listbox object

Description ASCII code of the character that designates column breaks in Items.

Double click Opens character table. Double click on the character to use.

Settings 0 to 255

Details For multi-column objects, when you add a line of text to the object, Everest scans it for the ColChar character to know where to skip to the next column.

Examples The following example adds an item to a multi-column Listbox that employs the comma (ASCII 44) as the column designator:

```
Listbox(1).ColChar = 44
Listbox(1).AddItem = "Number of boxes,61"
```

The following example retrieves the contents of the Listbox(1) "cell" at line 7, column 3 (which is the fourth column from the left, because the leftmost column is column 0):

```
getlyn = 7: getcol = 3
Listbox(1).LookAt = getlyn
contents = pik(getcol+1, Chr(.ColChar) + .Item)
```

Notes Changes in the ColChar attribute might be reflected only in subsequently added items.

Also see Divider, Pik() function

ColCount Attribute

Applies to Listbox object

Description Use this attribute to obtain multiple columns.

Settings

0 to 49	desired number of equal width columns
30,40,50	(example) pixel width of (3) columns
-30,-70	(example) percentage width of (2) columns

Details Setting this attribute to a single number from 0 to 49 causes Everest to reset the column widths equal to the Width of the object divided by the number of columns.

If you want columns of differing widths, set ColCount to a series of numbers separated by commas. Use positive numbers to specify the column widths in pixels. Use negative numbers to specify the column widths in percentages of the Width of the object.

To see the items arranged into your columns, be sure to include the ColChar character within the ItemList items. Use Divider to choose a visible column separator.

At run time, when the user clicks the mouse on the object, Everest determines which column was clicked and stores that information in Sysvar(177); the leftmost column is 0. To determine which row was clicked, use ItemIndex.

Examples The following example setting for ColCount divides the object into four columns, the largest of which is the second:

```
-20,-40,-20,-20
```

The following A-pex3 programming example retrieves the contents of the Listbox(1) "cell" at line 7, column 3 (which is the fourth column from the left, because the leftmost column is column 0):

```
getlyn = 7: getcol = 3
Listbox(1).LookAt = getlyn
contents = pik(getcol+1, Chr(.ColChar) + .Item)
```

Notes Presently, there is no difference between the settings of 0 and 1; for now, use 0 instead of 1.

If you change the Width of the object, you should reset (force an update of) ColCount if you want Everest to update the column widths.

Also see ColChar, Divider, Pik() function

COLOR Command

Applies to A-pex3 programming

Description Sets the color of Xgraphics, the window or Everest's 16-color palette.

Syntax COLOR (Which [, Red] [, Green, Blue])

Details The COLOR command controls the display color of several items. Most frequently, it precedes an Xgraphics drawing command in an A-pex3 program in order to set the color of the drawing. For a discussion of how Windows handles colors, see the Notes section of this topic.

Red, Green and Blue each range from 0 to 255. Use one of the following numbers for Which:

0 set background color

1 to 15 set Everest's palette colors

-1 set foreground color of subsequent Xgraphics for which you do not include a Color parameter

-2 set fill color of subsequent painted Xgraphics that have a non-transparent fill style (set via the STYLE command)

-3 same as 0, except employs palette colors; include desired color number (1 to 15) as Red parameter

-4 same as -1, except employs palette colors; include desired palette color number (0 to 15) as Red parameter

-5 same as -2, except employs palette colors; include desired palette color number (0 to 15) as Red parameter

-6 same as -3, except employs default 16 colors (same as those in Summit for DOS)

-7 resets Everest's 16-color palette back to default colors (same as those in Summit for DOS)

There are basically two methods for choosing colors for Xgraphics.

METHOD 1: SET FOREGROUND COLOR

With this method, you select a foreground color before drawing Xgraphics. This example draws two purple lines:

```
COLOR (-1, 255, 0, 255)
LINE (0, 0, 50, 50)
LINE (50, 50, 100, 0)
```

Any other Xgraphics commands that follow will also be drawn in purple, unless you insert another COLOR command to change the color or use Method 2.

METHOD 2: USE PALETTE COLOR

With this method, you assign up to 16 different colors to Everest's palette, and refer to them via the Color parameter of the Xgraphics command. This example draws a line in color 13 and one in color 14:

```
LINE (0, 0, 50, 50, 13)
LINE (50, 50, 100, 0, 14)
```

This example assumes you have previously stored the desired colors in Everest's palette, or find the default values acceptable. Everest automatically initializes its 16-color palette to match the typical colors used in DOS. These colors are:

Slot #	Color
0	black
1	blue
2	green
3	cyan
4	red
5	purple
6	brown
7	white
8	gray
9	bright blue
10	bright green
11	bright cyan
12	bright red
13	bright purple
14	yellow
15	bright white

To change the colors in the palette, use the COLOR command with a value of 1 to 15 for the Which parameter. The following example assigns a custom mixture of red and green into palette slot #6:

```
COLOR (6, 191, 127, 0)
```

Subsequent Xgraphics commands that include a Color parameter of 6, such as:

```
LINE (0, 0, 50, 50, 6)
```

will employ the custom red-green color mixture.

Everest stores the 16-color palette values in Sysvar(30) to Sysvar(45).

Notes **HOW WINDOWS HANDLES COLORS**

When you run Windows, you choose its display mode from the Control Panel. For example, you might run Windows in a 640 x 480 x 16 resolution. The last number represents the number of different colors Windows can display simultaneously; in this example, that is 16. On more powerful computers, Windows can display 256, 32768 or even more different colors simultaneously.

The actual colors that appear in that 16 or 256 color palette are determined by Windows, which in turn is influenced by the bitmapped pictures displayed on your page.

When a picture needs a color such as dark blue, Windows automatically finds the closest color in its palette, and uses that. If nothing is close, it will add dark blue to its palette (assuming the color being replaced is not needed elsewhere on the page...and sometimes even when it is).

THE RGB MIX

The maximum number of different colors Windows can display (as of this writing) is 16,777,216 (which is 256 * 256 * 256). That's the total number of possible combinations of 256 shades of each of red, green and blue. Red, green and blue are the three primary colors computers often mix to produce the full spectrum.

MIXING COLORS WITH THE COLOR COMMAND

In Everest, you can use the COLOR command to select from the 16,777,216 colors. You simply specify the amount of red, green and blue (RGB) you want to mix together. You use a number from 0 (none) to 255 (maximum) for each of the three colors.

Even if the computer hardware does not support all 16+ million colors, you still employ RGB color numbers in the 0 to 255 range. Windows automatically scales your color numbers into a range supported by the hardware.

Some examples: white is a combination of all 3 primary colors. If you set red, green and blue to 255, you will get the brightest possible white. If you use 128 in place of 255, you will get a less bright white (half as bright). Purple is a combination of red and blue. So, if you set red and blue to 255, and leave green at 0, you will get bright purple.

WHAT WINDOWS DOES TO COLOR

So, COLOR (Which, 255, 0, 255) will produce bright purple, right? Well, the answer is "it depends." If bright purple does not exist in Windows' palette, it will either create it (not likely) or choose the closest substitute (more likely). Sometimes Windows "dithers" colors to produce a substitute. Dithered colors are combinations of two or more other colors. They do not work well as backgrounds, but are usually acceptable as foreground colors.

There's no way to know for sure how Windows has produced the color. So, we recommend you stick to primary colors and simple combinations for best appearance on a wide variety of computers. Or, to make Windows choose the closest solid color, employ a negative number for the Reg, Green and/or Blue parameters, for example: COLOR (Which, -255, 0, -255).

FORCE WINDOWS PALETTE

Microsoft says that you can force the Windows color palette to contain certain colors by loading a .DIB file (that contains the desired colors) into a Picture object.

Also see BackColor, Rgb() Function, STYLE

Columns Attribute

Applies to	PicBin object
Description	Sets the number of columns into which to divide the image loaded into the Picture Bin.
Example	If the image in the Picture Bin is 640 pixels wide, and the width of each icon in the image is 32 pixels, you would set the Columns attribute to 20 (640/32).
Also see	PicBin Object, Rows

Combo Object

Description The Combo object is an enhanced Input object that allows users to make a selection from a list.

Attributes
AddItem
AdjustResponse
Answers1
Answers2
AntIncorrect1
BorderColor
Bottom
ClickEvent
CMIData
Comment
Condition
Create
Destroy
DragMode
EdgeSizeInner
EdgeStyleInner
Enabled
FillColor
FindString
Font3d
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
FoundIndex
GotFocusEvent
Height
IDNumber
Ignore
Initially
Item
ItemColor
ItemCount
ItemIndex
ItemList
Judgment
LastAdded
Left
LightColor
LookAt
LostFocusEvent
MaxDrop
MouseLeaveEvent
MouseOverEvent

MousePointer
Move
Name
Preset
RemoveItem
Right
SaveAsObject
SetFocus
ShadowColor
Sorted
Style
TabOrder
TabStop
Text
Top
Tries
Update
Visible
Width
Zev
ZOrder

Also see Input Object

Command Attribute

Applies to	Media object	
Description	Sends command strings to a MCI (multimedia) device.	
Settings	Open	prepares the DeviceType for further Commands
	Close	releases memory associated with keeping DeviceType open
	Play	activates DeviceType for playback
	Pause	pauses playing or recording on DeviceType; or, resumes if already paused
	Stop	ends playback or recording on DeviceType
	Back	steps backward on DeviceType
	Step	steps forward on DeviceType
	Prev	goes to the beginning of the current track, or if already within 3 seconds of the beginning of the track, goes to the beginning of the previous track
	Next	goes to the beginning of the next track
	Record	records on DeviceType
	Eject	ejects the media in DeviceType

Details

If the DeviceType is not already open, and you issue a Command (such as Play) that needs it to be open, Everest automatically opens the device for you.

At run time, numeric error codes are returned in the Sysvar(1) variable. A value of 0 in Sysvar(1) indicates no error.

Examples

To play a .WAV file, first set the Media object's DeviceType attribute to WaveAudio and the FileName attribute to the name of the desired .WAV file, then set the Command attribute to Play.

If you want the .WAV file to play repeatedly, set the Media object's DoneEvent to (substitute the appropriate IDNumber for 1):

```
Media(1).Command = "prev": Media(1).Command = "play"
```

If you want to immediately stop play of this .WAV file when the user clicks a Button, set that Button's ClickEvent to:

```
Media(1).Command = "close"
```

Also see

DeviceType, Mci() Function, Wait

Commands

Applies to A-pex3 programming

Description Commands are used in A-pex3 programs to issue instructions. Most authors enter commands by placing them in Program objects. Commands that fit on one line can also be placed in xxxEvent attributes. Because Everest executes commands only at run time, you can observe their effect only by running your page(s).

For more information about a particular command, refer to its entry elsewhere in this reference.

BRANCHING

BRANCH	go to another page
CALL	execute another page as a subroutine
GOSUB	execute a Program object as a subroutine
GOTO	redirect Program execution to a LABEL
JUMP	redirect page execution to a JLabel Object
LABEL	mark the destination of a GOTO
OPEN	display a page in another window
RETURN	exit from a subroutine executed via CALL

CONDITIONAL

ELSE	otherwise clause in IF block
ELSEIF	test another condition in an IF block
ENDIF	end an IF block
IF	test a condition
THEN	mark the end of the condition in an IF or ELSEIF

LOOP

DO	begin loop structure
LOOP	mark the end of a loop structure
OUTLOOP	exit from a loop structure
RELOOP	go back to the start of a loop structure

XGRAPHICS

ARROW	draw an arrow
BOX	draw a rectangle
CIRCLE	draw an ellipse
COLOR	set color of graphics
FBOX	draw a filled rectangle
FONT	choose font of subsequent PRINT text
GFILL	fill with graduated color
LINE	draw a line
LPRINT	send text to the printer
PAINT	fill an enclosed area with a color
POINT	draw a dot

POLY	draw a closed polygon
PRINT	display text
RBOX	draw a box with rounded corners
SCALE	customize window coordinate system
STYLE	set various graphics appearance attributes

OTHER

DELVAR	delete variable or array
DIM	allocate an array
DPRINT	display text in Debug window
ERASE	remove objects and/or graphics from window
PAUSE	wait for a timer period
REDIM	change the number of elements in an array
STEP	engages/disengages debug step mode

Also see Attributes, Functions, Operators, Program Object

Comment Attribute

Applies to	All objects
Description	Most authors enter a remark about or a description of the object in the Comment attribute. Available only at design time.
Details	The Comment appears at the bottom of the Book Editor window to document the purpose of the object in the page. To modify the comment, double click on it.

CommentAction Attribute

Applies to	Wait object
Description	Specifies the action to perform when the CommentActivator event is triggered.
Double click	First: sets CommentAction to BRANCH @comment. Next: Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.
Details	When a Wait object sees that an event code matches the CommentActivator event, it traps that event code, and performs the CommentAction. Most authors employ the CommentActivator and CommentAction to allow the user to write a comment about the page. To employ Everest's built-in comment system, enter `BRANCH @comment` as the CommentAction. A branch to @comment displays EVEREST.MSG message number -31 in a window, collects the user's response, saves it in the comment file (whose name is USER.ECM, unless you specify otherwise via the CommentFile entry in the EVEREST.INI file), and resumes where left off.
Notes	Everest's built-in user comment system can be disabled globally by deleting the text of EVEREST.MSG number -31.
Also see	CommentActivator, Wait Object

CommentActivator Attribute

Applies to	Wait object
Description	Specifies the numeric event code that triggers the CommentAction.
Settings	-32000 to 32000, or a string surrounded by quotes
Double click	Opens event code dialog box. Press the desired key to automatically generate the corresponding event code.
Details	Everest watches the events that occur in your project, and checks if one matches the event code you specify as the CommentActivator. If a match is found, the event is removed from the queue, and Everest performs the CommentAction.
Example	To make a Ctrl+C keypress the event that invokes the CommentAction, set the CommentActivator to the event code for Ctrl+C: 2067.
Also see	CommentAction, Wait Object

Condition Attribute

Applies to — Animate, Button, Check, Combo, Erase, Flextext, Frame, Gauge, HScroll, Input, Layout, Line, Listbox, Mask, Media, Menu, OLE, Option, PicBin, Picture, Program, Shape, SPicture, Textbox, Timer, VScroll, Wait objects

Description — Controls whether an object is added to, updated, or removed from the window. Read-only at run time.

Settings

-1	Create/update object (default). If an object of the same class and the same IDNumber already exists in the window, replace it with this new one. If no such object already exists in this window, add this new object to the window.
0	Remove/do nothing. If an object of the same class and same IDNumber already exists in the window, disable it and make it invisible (erase it). Otherwise, do nothing.
1	Create only. If there is no object of the same class and IDNumber in this window, add this new object to the window. Otherwise, do nothing.
2	Ignore object. Skip past it when running the page.
$$	Ignore object. Skip past it when running the page. Everest uses this code when you toggle off an object in the Book Editor window.
IF...	Conditional (see below).
var	The name of a variable that contains one of the numeric codes above.

Details — As Everest runs your page, it executes each object one-by-one in the order in which the objects appear in the page. When executing an object, Everest first checks the Condition attribute, and takes one of the actions described above.

The value of the Condition attribute can be expressed via a numeric constant, variable, or IF expression.

Authors frequently use the Condition attribute to display one of several feedback messages. This can be done by entering an IF expression as the Condition.

To employ a variable, simply enter its name. Do not enclose the name in brackets. Everest checks the numeric value contained in the variable at run time.

Example — To use an IF expression, type the word "if" followed by the conditional expression. For example:

```
IF response = "answer"
```

NOTE: leave off the "THEN" portion. If the condition is true, Everest performs a "-1 Create/update" action. If the condition is false, Everest performs a "0 Remove/do nothing" action. It is not possible to obtain other actions with an IF expression.

Also see IDNumber, IF

ControlBox Attribute

Applies to	Layout object
Description	Determines whether a Windows control box is displayed in the upper-left corner of the window.
Settings	Yes display control box No do not display control box
Details	The Windows control box is found on most windows. Among other actions, it allows the user to move and close the window. A control box is available only when the window's TitleBar is also enabled.
Also see	CloseEvent, MaxButton, MinButton, MoveEvent, ResizeEvent, TitleBar

CopyBgnd Attribute

Applies to Picture object

Description Copies the image behind the Picture object into the Picture (at run time only).

Settings

Yes	copy the image
No	do not copy (i.e. do nothing)
1	copy the image later (such as during CopyPic), available at run time only

Details CopyBgnd is handy for creating a Picture object that is seemingly transparent. Most authors use CopyBgnd to create removable text and/or graphics on top of an image or pattern that is in the background of the window. Authors often overlay graphics (via Xgraphics commands, DrawText or TpColor) onto the Picture, and remove them easily when done (by erasing the Picture object).

To observe CopyBgnd in action, perform the following steps:

1) Start a new page (choose New from the File menu)
2) Drag in a Layout object.
3) Set the BgndPicture attribute of the Layout object to the name of a graphics file.
4) Drag in a Picture object; place it anywhere on top of the BgndPicture.
5) Leave the CopyBgnd attribute set to No.
6) Drag in a Wait object.
7) Now, run a Preview of the page. You'll observe that the Picture object obscures a rectangular area of the BgndPicture.
8) Stop the Preview and return to editing.
9) Set CopyBgnd to Yes.
10) Run a Preview again. The Picture object no longer obscures the BgndPicture. In fact, it seems the Picture object is not even there...it has become transparent. It is ready for additional graphics to be drawn on it.

Notes If CopyBgnd does not seem to copy the desired image, enable AutoRedraw for the Layout object.

CopyBgnd employs the Windows BitBlt function, and is subject to its limitations.

During editing, CopyBgnd does not copy the background image. Be sure to run your page(s) to check for proper operation.

If you enable CopyBgnd (set it to -1) at run time via A-pex3 programming, Everest updates the Picture object with the current background image.

Also see AutoRedraw, CopyPic, DrawText, TpColor

CopyPic Attribute

Applies to Picture object

Description Copies an image from one Picture object to another, or from a window. Write only. Available at run time only.

Settings

> 0	the IDNumber of the Picture object from which to copy
0	the current window
< 0	a specific window number (example: for window 1 use -1)

Details Since CopyPic will display an image quickly, authors often pre-load a large image into a hidden Picture object (to save time), then copy the image to a visible Picture object as needed.

When copying between Picture objects, CopyPic respects the CopyBgnd and TpColor attribute settings of the destination Picture. Therefore, if CopyBgnd is enabled for the destination, CopyPic first updates the background, then copies the image. If TpColor is in use for the destination, that color will be transparent in the copied image. These features are handy for custom animation (see the example below).

If the destination Picture employs a SpecialEffect, and the source Picture is not visible in the window at the time CopyPic is used, only the image loaded into the source Picture while its AutoRedraw attribute is enabled will be copied. (If CopyPic seems to fail, try enabling AutoRedraw for the source Picture.)

If the destination Picture employs a SpecialEffect, upon execution of CopyPic, Everest must enable the source Picture's Visible attribute. If you do not want the source Picture to be seen in the window at run time, position it off the edge (for example, set its Left attribute to 10000).

If the destination Picture is not completely visible within the window, and CopyPic appears to fail, try enabling its AutoRedraw attribute. Alternatively, try invoking a refresh via the destination Picture's Update attribute.

Example The following example copies an image from the Picture object with IDNumber 1 to the Picture object with IDNumber 2:

```
Picture(2).CopyPic = 1        $$ source is 1, dest is 2
```

The following example demonstrates a custom animation technique (moving three images across the window from left to right). To try this example, place four Picture objects on a new page, followed by a Program object that contains:

```
$$ first, prepare 3 off-screen source images
Picture(1).AutoRedraw = -1
.Left = 11000
.PictureFile = "step1.bmp"

Picture(2).AutoRedraw = -1
.Left = 12000
.PictureFile = "step2.bmp"
```

```
Picture(3).AutoRedraw = -1
.Left = 13000
.PictureFile = "step3.bmp"

$$ prepare destination
Picture(4).CopyBgnd = 1
Picture(4).TpColor = rgb(255,255,255)     $$ white

$$ loop for animation
iter = 0
DO
  Picture(4).Left = iter
  Picture(4).CopyPic = 1
  Picture(4).CopyPic = 2
  Picture(4).CopyPic = 3
  iter++
LOOP IF iter < Window(0).Width
```

Also see AnimPath, AutoRedraw, CopyBgnd, SpecialEffect

Cos() Function

Applies to	A-pex3 programming
Description	Returns the trigonometric cosine of an angle.
Syntax	cos(Angle)
Details	Express Angle in radians. To convert from degrees to radians, multiply by (pi/180).
Example	The following example stores the cosine of a 45 degree angle in the variable named myangle:

```
myangle = cos(45 * 3.141593 / 180)
```

Also see Atn() Function, Sin() Function, Tan() function

Create Attribute

Applies to — Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects

Description — Adds an object to the window at run time.

Details — Normally, objects are added to the window as they are encountered in a page. Experienced authors can use the Create attribute to add objects at run time via A-pex3 programming. Read-only.

Example — To use Create, simply place it on the right side of a variable assignment in a calculation. The following statement adds a Checkbox with IDNumber 30 to the window:

```
ok = Check(30).Create
```

If an error occurs during the creation process, the variable on the left side (named "ok" in the example above) is set to the numeric error code, otherwise it is set to 1.

Typically, you also need to Enable the object and make it Visible after it has been created. For example:

```
IF ok = -1 THEN   $$ -1 means no error
  Check(30).Enabled = -1
  Check(30).Visible = -1
ENDIF
```

By default, Everest locates the object near the center of the window. To adjust the location and size, use the Move attribute.

Notes — Create can only be used with object classes that allow more than one object in a window. For other objects, use the Enabled attribute.

Also see — Destroy, Enabled, IDNumber, Obj() Function, Zev

Cvi() Function

Applies to A-pex3 programming

Description Converts a 2-byte string created via the Mki() function back into a numeric value.

Details Cvi() is the converse of Mki(). In terms of an analogy, Cvi() is to Mki() as Asc() is to Chr().

Example The following example sums 10 random numbers stored in the variable named packed via Mki() (see the example for Mki()):

```
count = 1: sum = 0
DO
  sum = sum + cvi(packed $- (count * 2 - 1))
  count++
LOOP IF count <= 10
```

Also see Asc() Function, Mki() Function

Dat() Function

Applies to A-pex3 programming

Description Returns calendar date information.

Syntax dat(Operation)

Details The Dat() function returns different values based upon Operation:

When Operation is	Dat() Returns
""	today's date in MM-DD-YYYY format
0	today's date in MM-DD-YYYY format
1	current month number (1 to 12)
2	current day number (1 to 31)
3	current year number (1900 to 2000)
4	current year number (00 to 99)
5	current day of week number (1 = Sunday, 7 = Saturday)
6	number of days since December 30, 1899 12:00:01 AM

USER-DEFINED DATE STYLES

Employ the following characters in the Operation parameter to create your own date formats. Be sure to surround the characters with quotes.

d	day of month as a number without leading zero (1 to 31)
dd	day of month as a number with leading zero (01 to 31)
ddd	day of week name as an abbreviation (Sun to Sat)
dddd	day of week name (Sunday to Saturday)
ddddd	date in Short Date form
dddddd	date in Long Date form
m	month as number without leading zero (1 to 12)
mm	month as number with leading zero (01 to 12)
mmm	month name as an abbreviation (Jan to Dec)
mmmm	month name (January to December)
q	quarter of the year as a number (1 to 4)
y	day of year as a number (1 to 366)
yy	year as a two digit number (00 to 99)

yyyy year as a four digit number (1000 to 9999)

DAY NUMBER

The dat() function can also return for any date the number of days elapsed since December 30, 1899. Express the desired date in MM-DD-YYYY format in the Operation parameter. This feature can be used to determine the number of days between any two dates (see example below).

Examples The following A-pex3 program puts today's date in the variable named today:

```
today = dat(0)
```

The following example displays the current month name in the TitleBar of the current window:

```
Window(0).Caption = dat("mmmm")
```

The following example calculates your age expressed in number of days:

```
birthday = ibx("Enter your birthdate in MM-DD-YYYY format")
IF birthday =P= "##-##-####" THEN   $$ check format
  days = int(dat(6) - dat(birthday))
  dummyvar = mbx("You are " + days + " days old", 48)
ELSE
  dummyvar = mbx("That was not MM-DD-YYYY format", 16)
ENDIF
```

Also see Fmt() Function, Tim() Function

DblClickEvent Attribute

Applies to	Animate, Flextext, Listbox, OLE, Picture, SPicture
Description	Event code to generate, or programming to perform, when the user clicks the mouse twice while pointing to the object.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	When you enter a number or string constant for DblClickEvent, you are merely telling Everest what event to generate when the user clicks twice on the object. To make use of that event (i.e. detect it and do something useful, like BRANCH to a menu), you must include a Wait object in your page.
Also see	ClickEvent, Wait Object

Dde() Function

Applies to A-pex3 programming

Description Performs various Windows Dynamic Data Exchange (DDE) operations. Intended for use by experienced DDE programmers.

Syntax dde(Operation [, Item])

The Operation parameter is a number that specifies the operation to perform, as follows:

0	set LinkMode to Item
1	set LinkTopic to Item
2	set LinkItem to Item
3	set LinkTimeout to Item
4	poke Item to server
5	request and return data from server
6	return last data from server
7	send Item to server for execution

Item is a character string; not all Operations have an Item.

Details DDE is a means by which you can obtain information from other currently running Windows applications, and provide information to them. Applications that supply infomation are known as "servers" and applications that receive information are known as "destinations."

A complete discussion of DDE is beyond the scope of this manual. A good source of details is the Microsoft Visual Basic Programmer's Guide. The discussion here uses the same terminology as in that guide. For an example of DDE in use, see Everest's EDLLDRV.BAS program (which drives the Dll() function).

OPERATION 0 - SET LINKMODE

Set LinkMode after setting LinkTopic and LinkItem. Use one of the following values for Item:

0	end DDE link (if any)
1	establish automatic link
2	establish manual link

In an automatic link, the server sends the information specified by LinkTopic and LinkItem any time it changes. You can then retrieve the information in your Everest project via Dde() Operation 6.

In a manual link, you request the server to send the information specified by LinkTopic and LinkItem only when you need it. You do so via Dde() Operation 5.

OPERATION 1 - SET LINKTOPIC

LinkTopic specifies the name of the application with which you want to establish a DDE

link, as well as the topic of discussion. Separate these two pieces of information with a | (ASCII 124) character.

Valid settings for LinkTopic depend on the other application; consult its documentation. For example, a valid DDE LinkTopic for Microsoft Excel might resemble:

```
ecode = dde(1, "Excel|C:\excel\4q93.xls")
```

To establish a DDE link from another application into your Everest project, use the name of the Everest EXE that is running, plus the word Variables. For example: "STUDENT|Variables".

OPERATION 2 - SET LINKITEM

LinkItem specifies the exact item of the DDE conversation. For example, in a DDE link with Microsoft Excel, you might set LinkItem to the name of a spreadsheet cell in order to obtain the contents of that cell.

Valid LinkItems depend on the application with which you are communicating. Consult its documentation to learn what LinkItems it supports.

If your project is acting as a server, the destination should set its LinkItem to the word Request.

OPERATION 3 - SET LINKTIMEOUT

Applications vary in the amount of time they need to respond in DDE conversations. If an application takes too long to respond, Everest returns error 286. You can specify how long to wait via LinkTimeOut. Specify LinkTimeOut in tenths of seconds.

OPERATION 4 - POKE TO SERVER

The destination in a DDE conversation can also send information to the server. This is called poking the data to the server. Consult the server application's documentation to determine what to poke.

If your Everest project is acting as a server, the destination can obtain information from it by poking the request. To obtain the value of a variable in your project, the destination should poke the name of the variable surrounded by { }.

OPERATION 5 - REQUEST & RETURN INFORMATION

For manual links, where your project is acting as the destination, you obtain information from the server by requesting it when needed. For example:

```
getinfo = dde(5)
```

requests the lastest information from the server, and returns it.

OPERATION 6 - RETURN INFORMATION

For automatic links, it is not necessary to request updated information from the server. You simply need to retrieve that information into your project. For example:

```
getinfo = dde(6)
```

OPERATION 7 - SEND EXECUTE STRING

Operation 7 sends a command to the other application for execution. The types of commands recognized and their syntax depend on the other application. For example, you might be able to tell the other application to Save any changes by doing:

```
ecode = dde(7, "Save")
```

Your Everest project automatically responds to execution strings sent by other applications. Execution strings sent to Everest should have the form of A-pex3 commands. For example, if you send the execution string BRANCH mainmenu, Everest will perform that instruction just as if it had been entered as part of the project. In theory, you could write an external application that drives your entire Everest project.

OPERATION 8 - SEND INFORMATION

Use this operation to leave information for the other application to obtain whenever it requests it. Set Item to the information to leave.

ERROR CODES

Numeric error codes are returned by the Dde() function, except for Operations 5 and 6. A value of -1 indicates no error. For Operations 5 and 6, the error code is returned in Sysvar(1).

Example

The following A-pex3 program attempts to establish a link with the Microsoft Excel spreadsheet named 4Q95.XLS, and obtain the value in cell R1C1:

```
tries = 0
LABEL retry
tries = tries + 1
ecode = dde(0, 0)
ecode = dde(1, "Excel|C:\excel\4q95.xls")
ecode = dde(2, "R1C1")
ecode = dde(0, 2)
IF ecode = 282 & tries < 3 THEN
  ecode = shl("Excel")
  GOTO retry
ELSEIF ecode # -1 THEN
  ecode = mbx("Error " + ecode + " during DDE.")
  cell = ""
ELSE
  cell = dde(5)
ENDIF
```

Notes

If the application with which you are attempting to establish a DDE conversation is not already running, the Dde() function returns error code 282. You can detect this and use the Shl() function to start it.

Your Everest project can engage in only one DDE conversation at a time.

Also see OLE Object, Shl() Function

Debug Window

The Debug Window is accessible from the main Author window's Window pull-down menu. Authors use the Debug Window to help determine the cause of a execution problem in a page and/or book. Typically, the Debug Window is needed only to solve difficult problems.

OPENING THE DEBUG WINDOW

To manually open the Debug Window, choose it from the pull-down menu named "Windows." To automatically open it at run time, use the STEP command.

EXAMPLE OF DEBUG USE

The Debug Window is handy if you have been unable to determine why an object or variable gets set to a certain value. As an example, imagine you are displaying the user's exam score in Textbox(3), and for some reason, the Textbox displays "1" when it should display "100." The Debug Window can help you find where the Textbox get set to "1." This is described below.

BREAKPOINTS

In the Debug Window, you can set a breakpoint. A breakpoint is an expression that, when true, will cause Everest to stop the test run of your book. To set a breakpoint, double click on the breakpoint line in the Debug Window. Everest will ask you to enter an expression. To test the example above, you would enter:

```
IF Textbox(3).Text = "1"
```

Then, run your book. As soon as the expression becomes true, Everest will put your book into "step mode" (explained below) and display the Debug Window. Information in the window will tell you the current page, object and A-pex3 programming instruction. From this you should be able to isolate the cause of the trouble.

STEP MODE

Sometimes it is difficult to find the cause of a problem because your book runs too fast. Perhaps something flashes on the screen, then is gone. The Debug Window's step mode feature can help. It executes your book one piece at a time, pausing after every object and every line of programming code. This can give you time to investigate the trouble.

To engage step mode, choose it from the Debug Window's pull-down menus. While in step mode, the Debug Window will pop up repeatedly. You can press F2 to perform the next step, or use the pull-down menus to inspect the values in variables, etc. To disengage step mode (i.e. return to normal execution), press F4.

STEP UPON ERROR

Another feature activates step mode only if an error (something displayed in the "Uh oh" window) occurs while running your project. Use the "Step upon error" feature in the pull-down menus. This is handy when most of your project runs correctly, and you don't want to step through a long sequence of pages to get to a repeatable error.

WATCH EXPRESSION

The watch expression feature lets you observe the value of a variable, attribute or expression as your book executes. Everest updates the watch results with every calculation your book makes. For example, to watch the per cent of free Windows system resources as your book executes, double click on Watch Expression and enter:

```
fre(0)
```

Then run your project. If your project's window obscures the Debug Window, you may want to enable the "Always on top" feature in Debug's pull-down menus.

DPRINT TEXT

Place DPRINT commands in your programming, and the results appear in the Debug Window. This can be helpful to track what pages in your book are being executed.

EXECUTION SPEED

You will probably notice that the execution speed of your book decreases when the Debug Window is open. This is because Everest has to do a fair amount of work to update the contents of the Debug Window. To restore the original execution speed, close the Debug Window.

DELVAR Command

Applies to A-pex3 programming

Description DELVAR removes a variable or array from memory, releasing the space it occupies.

Syntax DELVAR <VarName>
or
DELVAR (clear)

Details DELVAR is most frequently used with arrays to clear them from memory.

DELVAR (clear) deletes all author defined variables; system variables are left unchanged.

Example The following example removes the array named scores and immediately reallocates it with 100 elements:

```
DELVAR scores
DIM scores(100)
```

Notes Relatively speaking, DELVAR is a slow command. When you delete a variable, Everest must shift other variables to reclaim the memory that had been occupied. That's a fair amount of work, so don't use DELVAR in speed critical sections of your project.

To include other A-pex3 commands following DELVAR on the same line, separate with a colon AND a space. For example:

```
DELVAR scores: DIM scores(100)
```

Also see Arr() Function, DIM, REDIM, Var() Function

Destroy Attribute

Applies to	Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Line, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects
Description	Removes a particular object from a window, or closes a window. Read-only.
Details	Destroy removes one object from a window at run time. Compare this with the Erase object and ERASE command which remove one or more objects from a window. Destroy is intended for use by experienced authors.

Examples	To remove the Check object with IDNumber 30, use the following calculation:
	`ok = Check(30).Destroy`
	Numeric error codes (if any) are returned in the variable on the left side; if the object is removed successfully, -1 is returned.
	To close window number 2, use the following calculation:
	`ok = Window(2).Destroy`

Also see	CloseEvent, Create, ERASE, Erase Object, Obj() Function, OPEN

DeviceType Attribute

Applies to Media object

Description Specifies the name of the multimedia device to operate.

Settings

AVIVideo	Microsoft Video for Windows (.AVI file)
CDAudio	Compact disc audio
DAT	Digital audio tape
DigitalVideo	digital format video
MMMovie	Multimedia movie
Other	
Overlay	video overlay device
Scanner	input from scanning device
Sequencer	MIDI
VCR	Video cassette recorder
Videodisc	video and audio
WaveAudio	digitized sound (.WAV file)

Details When creating a new Media object, choose a DeviceType first. Next, if the desired media is stored in a disk file (such as a .WAV file or .AVI file), specify it via FileName. Finally, set the Command.

Everest operates the various DeviceTypes by communicating with their software drivers. The drivers are not supplied with Everest. Many drivers come with Microsoft Windows; others are provided by hardware manufacturers.

We have found problems in some drivers, particularly those for MMMovie. If you experience trouble, contact the driver's vendor to be sure you have the most up-to-date version. Due to the nature of the drivers, we cannot guarantee that Everest will work with every DeviceType.

Example Here's how to play an .AVI motion video file (Microsoft Video for Windows file) in your project:

1) Put an object into the page that can act as a container for the .AVI; a Picture object works well.

2) Put a Media object after the Picture object.

3) Set DeviceType to `AVIVideo`.

4) Set DisplayIn to `picture(1)`. The "1" is the IDNumber of the container object; your value might be different.

5) Set Command to `Play`.

6) Set FileName to the name of the .AVI file. You can double click on FileName to open the load file dialog box.

7) If you want to resize the video to fit the container, enable AutoScale.

8) Put a Wait object after the Media object.

9) Run a Preview of the page.

Also see DisplayIn, Mci() Function

DIM Command

Applies to A-pex3 programming

Description Explicitly allocates the requested number of elements in a single-dimensional array (provided the array does not yet exist).

Syntax DIM Arrayname(Elements)

Arrayname is the name of the array. Standard rules for variable naming apply to arrays as well.

Elements is the number of slots to reserve.

Details If the array already exists, Everest ignores the DIM command. To change the number of elements in an existing array, use the REDIM command. To determine if an array already exists, use the Var() Function at run time.

Arrays actually contain Elements+1 elements because they also contain an element numbered 0.

If you do not use DIM to create an array, it is created implicitly upon first use. The number of elements allocated matches the number of the element first used.

The best place for DIM commands is in a Program object with the special key name of @define. Before running a project, Everest looks for a Program object named @define in the book, and if it exists, executes it. Be sure to set the SaveAsObject attribute of an @define Program object to Yes.

Examples The following example creates an array named files containing 50 elements (actually, 51 if you count the 0th element):

```
DIM files(50)
```

The following example creates the array only if the page is not being run as a preview:

```
IF sysvar(54) # 0 THEN DIM files(50)
```

Notes If Arrayname is a single letter, Everest creates a minimum of 32 elements in the array; this is for compatibility with Summit for DOS.

Everest automatically initializes new array elements to null strings.

DIM does nothing if the array already exists. Use REDIM to change the number of elements in an existing array.

Also see Arr() Function, DELVAR, REDIM, Var() Function

DisableObjs Attribute

Applies to Judge object

Description Determines whether objects are automatically disabled after judging.

Settings
-1 judged objects are disabled
0 judged objects are not disabled
1 same as -1, except also disable if response matches Ignore

Details After Everest judges interactive objects, such as Check and Input, authors often want to disable those objects to prevent further user entry. When DisableObjs is set to -1, Everest automatically disables interactive objects when either the number of Tries is exhausted, or the user's response is judged as correct.

Notes A given object will not be disabled unless its Tries attribute is set to a number less than 10.

Everest disables an object by setting its Enabled attribute to No (0). You can reenable such an object via A-pex3 programming by setting Enabled to Yes (or -1).

Note that disabled objects (of certain classes) are "grayed out" per Windows standards.

Also see Enabled, Tries

DisplayIn Attribute

Applies to Media object

Description Specifies the object in which to display visual multimedia device output.

Details Certain multimedia devices, such as the AVIVideo and MMMovie DeviceTypes, display visual images. If you want to display these images at a particular location in the window, set DisplayIn to the name of an object that will act as a container. Any object with an hWnd attribute, such as a Picture or Button, can act as a container.

Example Most authors display .AVI video within a Picture object. Here's how:

1) Add a Picture object to your page; make sure it appears in the page before the Media object.

2) Edit the Media object. Double click on the DisplayIn attribute until the Name of the Picture appears (of course, you can type the Name directly if you prefer). This name might resemble: `page1_picture_A`.

3) Set the DeviceType attribute to `AVIVideo`. Set the Command attribute to `Play`. And, set the FileName attribute to the name of the .AVI file you want to display.

4) Finally, run a Preview of the page to view the results.

Notes The image is not resized to fit the container unless AutoScale is enabled, and Windows is able to scale the image.

To use the entire current window as the container, enter `window(0)` for DisplayIn.

If the object you specify in DisplayIn does not exist in the current page (i.e. it was added to the window at run time via a previous page), you must not use the Name; instead use the ClassName(IDNumber) syntax. For example, your DisplayIn setting might resemble `picture(1)`.

Also see AutoScale, DeviceType, Mci() Function, Picture Object

Divider Attribute

Applies to Listbox object

Description Determines the style of the divider used between both rows and columns of the object.

Settings

0	None
1	BorderColor
2	ShadowColor
3	Lowered
4	Raised

Notes Due to a bug in the MicroHelp control that drives this object, changing the BorderColor setting might not update the color of the dividers while Divider is set to 1.

Also see ColCount

Dll() Function

Applies to A-pex3 programming

Description The Dll() function calls external routines in the Windows API and custom .DLL files.

Syntax dll(RoutineName [, Parm1 [, Parm2 ... [, ParmX]]])

RoutineName is the name of the routine to call. If you use a string constant for RoutineName, be sure to surround it with quotes.

Parm1 through ParmX are optional parameters needed by RoutineName. Everest allows up to 16 parameters.

Details The Dll() function works by communicating with Everest's DLL driver module (the default name of which is EDLLDRV.EXE). The EDLLDRV.EXE module should be located in the same directory as the Everest program that is running (i.e. AUTHOR or ERUN).

As supplied with Everest, the EDLLDRV.EXE contains declarations for a few routines. You can add more by modifying the EDLLDRV.BAS file with Microsoft Visual Basic 3.0 for Windows. Or, we can add additional routines for you for a small fee. Contact technical support for details.

Here are the Windows API routines currently supported by EDLLDRV.EXE (consult a Windows Programmer's Reference manual for details):

GetDriveType
GetKeyboardType
GetSysColor

Example The following example calls the Windows API GetKeyboardType routine via the Dll() function to determine the number of function keys present on the user's computer:

```
fkeys = dll("GetKeyboardType", 2)
```

DO Command

Applies to A-pex3 programming

Description The DO command marks the start of a loop structure.

Syntax
```
DO [IF <Condition>]
  <Action>
LOOP [IF <Condition>]
```

Details Use the DO command when you want to perform an action one or more times. A <Condition> is optional. When <Condition> is included, and is not true, Everest bypasses the DO...LOOP and continues processing after the LOOP command.

The syntax of <Condition> is the same as that for the IF command.

<Action> is any legal A-pex3 calculation or command. Be sure to indent <Action> with at least one space.

Example The following example looks for the last letter "A" in a string by starting at the end of the variable named "string" and progressing backwards:

```
found = len(string)
DO IF found > 0
  IF string^^found = "A" THEN OUTLOOP
  found--          $$ faster than found = found - 1
LOOP
```

Notes Beware of infinite loops! If <Condition> is always true, and you do not employ an OUTLOOP command, Everest will continue to execute the DO...LOOP indefinitely.

If your program is stuck in an infinite loop, you can press the Break key to stop the loop.

DO...LOOPs can be nested up to 8 levels.

Also see IF, OUTLOOP, RELOOP, Timer Object

DoEvents Attribute

Applies to Erase object

Description Determines if and when your project yields to allow Windows to process pending events.

Settings
0 do not yield to process pending events
1 process pending events after Erase object is performed
2 process pending events before Erase object is performed
3 process pending events both before and after Erase object

Details Most authors leave DoEvents set to 0. However, other settings can sometimes help cure display flickering or hesitation as described below.

Most of the time, Microsoft Windows processes events (such as a keypress, or the loading of a picture file) in the order in which they are received. However, if Windows is busy at the time an event arrives, it holds the event in a queue, and processes it as soon as possible.

It is rare, but possible, that the queuing of events by Windows can influence the appearance of your project. For example, if Windows is busy, it may not immediately update the contents of your project's window when you branch from one page to another. You might possibly observe this as a flicker, or hesitation, when Everest adds or removes objects from the window.

The DoEvents attribute tells Windows to pause and process pending events. Often this can improve your project's appearance. You can experiment by changing the value of DoEvents, and re-running your project. If alternative settings of DoEvents make no difference, then restore the setting of 0.

Notes Use DoEvents at your own risk. Since DoEvents can change the order in which events are processed, certain timing or sequence sensitive operations could be disturbed, and might not function properly.

Also see Ext(101) Function, LockUpdate, Update

DoneEvent Attribute

Applies to — Media object

Description — Event code to generate, or programming to perform, when the Media object completes a Command.

Settings — -32000 to 32000, or a string surrounded by quotes
or
any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN

Double click — Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.

Details — If you would like to know when a Media object has completed a Command (for example, done playing an audio segment), enter a numeric event code for the DoneEvent attribute. Some authors use the DoneEvent to know when to start playing the next media element (or replay the current one).

You can detect the event, and take other actions, via a Wait object.

The DoneEvent places one of the following numeric values in Sysvar(1):

1	Command completed successfully
2	Command superceded by another
4	Command aborted by the user
8	Command failed

Example — The Media object with IDNumber 91 plays a .WAV file. To replay the .WAV upon completion, set DoneEvent to:

```
Media(91).Command = "prev": Media(91).Command = "play"
```

Notes — Due to a bug in Windows, the DoneEvent might not be generated if the Mbx() or Ibx() functions are in use at the time of the event.

Also see — Command, UpdateEvent

DPRINT Command

Applies to	A-pex3 programming
Description	Sends text to the Debug window for code debugging purposes.
Syntax	DPRINT (Text)
Details	While debugging your project, sometimes it is useful to know when certain portions of a program have been reached, or to display the contents of a variable. The DPRINT command sends Text to the Debug window for display.
Example	`DPRINT ("Counter variable = " + counter)`
Notes	For proper operation, you must include a space between DPRINT and (. To open the Debug window, choose it from the list in the Author window's Window pull-down menu.
Also see	PRINT, STEP

DragDropEvent Attribute

Applies to — Layout object

Description — Specifies the event code to generate, or programming to perform, when a user releases a mouse button to drop an object that had been dragged.

Settings — -32000 to 32000, or a string surrounded by quotes
or
any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN

Double click — Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.

Details — Use a Wait object to trap the event code you specify, and take the desired action. A common action is to set DragMode to 0.

When the user drops an object, Everest stores the object's class code number and IDNumber in the Sysvar(113) and Sysvar(114) variables, respectively. Everest also stores the class code number and IDNumber of the object on which it was dropped in the Sysvar(111) and Sysvar(112) variables, respectively. Object class code numbers can be found in the documentation for the Obj() Function.

Example — In this example, the Textbox with IDNumber 1 previously had its DragMode attribute set to 1. When the user drops the object (after dragging), a DragDropEvent is generated. To move the Textbox to the location at which it was dropped, set the DragDropEvent attribute of the Layout object to (all on one line):

```
Textbox(1).Move = reg(Textbox(1).Left + sysvar(9) - sysvar(159),
Textbox(1).Top + sysvar(10) - sysvar(160))
```

Also see — DragMode, Obj() Function

DragMode Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Listbox, Mask, Media, OLE, Option, Picture, SPicture, Textbox, VScroll objects

Description — Determines if the user can move the object within the window via the mouse.

Settings

0	not draggable
1	draggable

Details — When DragMode is set to 1, normal mouse handling of the object is disabled. Instead, the user can click on the object, hold down the mouse button, and drag it to a new location. When the user releases the mouse button, Everest generates a DragDropEvent.

You can detect the DragDropEvent via a Layout object, and take the desired action via a Wait object.

The location OF THE MOUSE POINTER when the user releases the button can be found in the Sysvar(9) and Sysvar(10) variables. The location of the mouse pointer just prior to the start of dragging can be found in the Sysvar(159) and Sysvar(160) variables.

Example — In this example, the Textbox with IDNumber 1 previously had its DragMode attribute set to 1. When the user drops the object (after dragging), a DragDropEvent is generated. To move the Textbox to the location at which it was dropped, set the DragDropEvent attribute of a Layout object to (all on one line):

```
Textbox(1).Move = reg(Textbox(1).Left + sysvar(9) - sysvar(159),
Textbox(1).Top + sysvar(10) - sysvar(160))
```

Notes — Due to a bug in Windows, the mouse cursor might change to a "not allowed" symbol when dragging over certain classes of objects.

Also see — DragDropEvent, MousePointer

DrawMode Attribute

Applies to Line, Shape objects

Description Determines how the drawing color (the pen) is combined with the display color (that already on the page).

Settings

1	Blackness Pen (always black)
2	Not Merge Pen (inverse of combination of pen color and display color)
3	Mask Not Pen (combination of the colors common to the display and the inverse of the pen)
4	Not Copy Pen (inverse of the color specified via BorderColor)
5	Mask Pen Not (combination of the colors common to both the pen and the inverse of the display)
6	Invert (inverse of the display color)
7	Xor Pen (combination of the colors in the pen and in the display, but not in both; applying twice restores the original)
8	Not Mask Pen (inverse of the combination of the colors common to both the pen and the display)
9	Mask Pen (combination of the colors common to both the pen and the display)
10	Not Xor Pen (inverse of the combination of the colors in the pen and in the display, but not both)
11	Nop (no operation; invisible)
12	Merge Not Pen (combination of the display color and the inverse of the pen color)
13	Copy Pen (normal drawing using BorderColor)
14	Merge Pen Not (combination of the pen color and the inverse of the display color)
15	Merge Pen (combination of the pen color and the display color)
16	Whiteness Pen (always white)

Details The effects of DrawMode are best observed by experiment. To see them, start a new page, add a Layout object, load a BgndPicture, add a Shape object on top of the background image, set BorderWidth to 5, then try the various DrawMode settings.

Also see BorderWidth, FillStyle, OutlineStyle

DrawPause Attribute

Applies to	Textbox object
Description	Controls the speed at which Text is drawn via DrawText.
Settings	0 draw the text without artificial delay > 0 number of seconds to pause after a line of Text (delimited via a Carriage Return character) is plotted < 0 same as > 0, except show an hourglass mouse cursor while drawing the Text
Details	Authors frequently use DrawPause to display a list of bulleted items with short timed pauses in between. When preparing the list of items in the Textbox, press the Enter key after each line where you want Everest to pause.
Example	To pause a half second between each CR-delimited line, set DrawPause to 0.5 and DrawText to -1.
Notes	If the DrawText setting indicates a special effect (i.e. if DrawText is set to a value greater than 0), the effect will be applied individually to each line of text.
Also see	DrawText, PAUSE

DrawShadow Attribute

Applies to	Textbox object
Description	Determines the color of the drop shadow (if any) for the DrawText Text.
Details	The drop shadow is displayed below and to the right of the text. Everest computes the distance of the drop shadow from the regular text based on the size of the characters. To override this distance, set Sysvar(199) to the desired distance (in number of pixels).
Notes	DrawShadow works only when DrawText is set to a value other than 0. Leave DrawShadow empty otherwise.
Also see	DrawText, PAUSE

DrawText Attribute

Applies to Textbox object

Description Controls whether the background of the Textbox is transparent. Can be modified at design time only.

Settings

-1	draw the Text as a graphic (with a non-destructive, transparent background) onto Picture objects or the BgndPicture
-2	same as -1, except temporarily enable AutoRedraw
0	plot the Text as an object (with a destructive, opaque background)
> 0	same as -1, except with a SpecialEffect

Details When DrawText is set to 0, a Textbox plots on top of underlying objects, thereby obscuring them. For example, when a Textbox is on top of a Picture object, the Textbox covers (obscures) a rectangular area of the Picture.

When DrawText is set to a non-zero value, at run time Everest instead draws the Textbox's Text (via a PRINT command), and makes the Textbox invisible. This makes the background of the Textbox transparent, thereby eliminating the obscuring rectangle.

Most authors enable DrawText when they want to display a small, transparent label on top of a Picture object or the window's Xgraphics or BgndPicture.

Due to a bug in Windows, in some situations when DrawText is set to -1, Windows might erase the text when refreshing the window. If you observe this problem (i.e. if your text disappears without reason), either change DrawText to a setting of -2, or enable the Layout object's AutoRedraw attribute.

Example While the setting of DrawText cannot be modified at run time, it can be reset. Resetting DrawText to itself tells Everest to replot the text of the Textbox. This is handy if you want to change the color of the text at run time. For example, an A-pex3 program to do so might resemble:

```
Textbox(1).ForeColor = rgb(255, 0, 0)      $$ red text
.DrawText = .DrawText
```

Notes At run time, if a user resumes where he/she left off, DrawText text is not redisplayed unless AutoRedraw is enabled for the object on which the text was plotted (i.e. for either the Layout or Picture object).

DrawText ignores the setting of Indent.

The effect of enabling DrawText is visible only at run time.

The transparent text drawing ability of DrawText works only with underlying Picture objects and the window (i.e. you cannot draw text onto SPicture, Button or other objects). Everest uses the upper-left corner of the Textbox to determine the object on which to draw the Text. If the corner of the Textbox is within a Picture, Everest draws the Text on that Picture. If multiple Picture objects overlap, Everest draws the Text on the one with the lowest IDNumber. If there are no underlying Pictures, Everest draws the Text onto

the window. Everest modifies the color and font attributes of the object or window on which it draws the Text.

The color of drawn Text may not be identical to that in the Textbox due to the way Windows handles color palettes.

To remove text plotted on a window via DrawText setting -1, use EraseType 1, 3 or 5, or load a new BgndPicture.

To remove text plotted on a window via DrawText setting -2, use EraseType 33 or 35.

Also see CopyBgnd, DrawPause, DrawShadow, PRINT, SpecialEffect, STYLE

EdgeDistInside Attribute

Applies to	Frame object
Description	Determines the distance between the outer and inner edges of the Frame.
Also see	EdgeStyleInside

EdgeSize Attribute

Applies to	Button, Check, Frame, Gauge, Option objects
Description	Determines the thickness of the outer edge of the object. The larger the setting, the more 3-dimensional the object appears.
Also see	EdgeStyle

EdgeSizeInner Attribute

Applies to	Combo, Gauge, Listbox objects
Description	Determines the thickness of the inner edge of the object. The larger the setting, the more 3-dimensional the object appears.
Also see	EdgeStyleInner

EdgeStyle Attribute

Applies to	Check, Frame, Gauge, Option objects	
Description	Determines the 3-D style used to draw the edge of the object.	
Settings	0	lowered
	1	raised
	2	chiseled
	3	shadowed right
	4	shadowed left
Also see	EdgeSize	

EdgeStyleInner Attribute

Applies to	Combo, Listbox object	
Description	Determines the 3-D style used to draw the edge of the object.	
Settings	0	lowered
	1	raised
	2	chiseled
Also see	EdgeSizeInner	

EdgeStyleInside Attribute

Applies to	Frame object	
Description	Determines the 3-D style used to draw the inside edge of the frame.	
Settings	0	lowered
	1	raised
	2	chiseled
Also see	EdgeDistInside	

ELSE Command

Applies to	A-pex3 programming
Description	Marks an "otherwise" portion of a block style IF command.
Details	Refer to the IF command.

ELSEIF Command

Applies to	A-pex3 programming
Description	Tests another condition in a block style IF command.
Details	Refer to the IF command.

Embedded File Manager Window

The Embedded File Manager Window is accessible via the main Author window's Utilities pull-down menu. The Embedded File Manager helps you keep track of files embedded in a book. For a comparison of external and embedded files, refer to Appendix F.

CROSS REFERENCE (Xref)

To use Xref, first highlight a file in the list on the left, and click Xref. The Xref feature searches the book to find pages that employ the file. This is a way to find old, unused files that are candidates for deletion. You can open a page for editing: simply double click on it in the Xref results list.

ADD FILE

Use the Add File button to embed or update a specific file to the book.

PEEK

Highlight a file, then use the Peek button to view it.

FRESHEN ALL

The Freshen All button updates the embedded in the book by recopying the original external files. This can be handy if you have modified the original, external files in the time since they had been embedded or last freshened.

DELETE

Highlight a file, then use the Delete button to remove it from the book.

Enabled Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, Option, Picture, SPicture, VScroll objects

Description — Controls whether an object is active and responsive to user interaction.

Settings
- Yes — object is active
- No — object ignores user interaction

Details — This attribute is set only from A-pex3 program code; it does not appear in the Attributes window. Refer to the Initially attribute for alternatives.

Example — To disable a Button (make it ignore clicks) use either of the following lines of A-pex3 programming code:

```
Button(id).Enabled = "No"
```

or

```
Button(id).Enabled = 0
```

where id is the IDNumber of the Button. Typically, Windows automatically "grays out" disabled objects to indicate their unavailability.

Also see — Condition, Initially, SetFocus, Tries, Visible, Zev, ZOrder

EndAt Attribute

Applies to Media object

Description Specifies the location at which to stop playing or recording a multimedia sequence.

Double click Opens the multimedia peek window.

Details This attribute is typically used to control the end point of a motion video clip, or the end point of CDAudio. When combined with the StartAt attribute, you can define the portion of the multimedia element to play.

You should express EndAt in the current TimeFormat. You can type the value of EndAt (if you know it), or peek at the media and have Everest generate the value for you.

MULTIMEDIA PEEK

To peek at the media, double-click on the EndAt line in the Attributes window. The Multimedia Peek window will appear with a row of buttons (Play, Stop, etc.). Everest activates only those buttons that are appropriate for the device. If all buttons are disabled, the DeviceType does not support peeking.

As the media plays, the Multimedia Peek window displays the current location in two formats: 1) as a 4-byte long integer Position, and 2) in the current TimeFormat as 4 individual values between 0 and 255, one for each byte, separated by colons.

The location values should be updated 10 times per second. We have found that some brands of multimedia hardware do not update the values this frequently. If you observe this problem, check with the hardware manufacturer to determine if you have the most current drivers.

When the media reaches the desired location, click either the "Use Position" or "Use TimeFormat" buttons. This copies the current corresponding location information to the EndAt attribute.

PLAY TO END

To play all the way to the end of the media, leave the EndAt attribute empty.

Also see DeviceType, StartAt, TimeFormat

EndFrame Attribute

Applies to	Animate object
Description	Controls the number of the frame at which the animation will stop.
Settings	0 no end frame > 0 frame number at which to stop
Details	If you set EndFrame to a number higher than the number of frames in the animation, the Animate object compensates by adjusting the number of Iterations.
Also see	Iterations, Position

ENDIF Command

Applies to	A-pex3 programming
Description	Marks the end of a block style IF command.
Details	Refer to the IF Command.

Env() Function

Applies to A-pex3 programming

Description Returns a DOS environment string.

Syntax env(Which)

Details The Which parameter can be a number or a character string. If Which is a number, the Env() function returns the corresponding string from the environment table. If Which is a character string (usually all upper-case letters), the Env() function returns the environment string with that name.

You can edit the environment table via the DOS SET command.

Example The following example stores the disk path to the DOS COMMAND.COM file in the variable named com:

```
com = env("COMSPEC")
```

EOFContinue Attribute

Applies to	Mask object
Description	Determines whether the focus moves to the next object in the tab order as soon as the Text attribute is filled with valid data.
Settings	Yes move focus automatically No do not move focus
Also see	TabOrder, TabStop

EOFEvent Attribute

Applies to	Input object
Description	Specifies the event code to generate, or programming to perform, when the user types at the end of an input field.
Settings	9 Tab to next object -32000 to 32000 event code quoted string event code or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	Note that Everest handles the value 9 in a special manner. For an EOFEvent of 9, Everest presses the Tab key as if the user did so manually. Typically, this moves the focus to the next object. To automatically initiate judging when the user reaches the end of an input field, set the EOFEvent code to the same code you use for JudgeActivator in the Wait object. Be sure to include a Judge object in the page as well.
Also see	JudgeActivator, TabOrder, TabStop

ERASE Command

Applies to A-pex3 programming

Description Removes objects and/or Xgraphics from the window.

Syntax ERASE (Action [, FromID#, ToID#])

Details The ERASE command performs the same actions as the Erase object; the command is provided for programming ease. Use one of the following numbers for the Action parameter:

0	do nothing
1	erase window background/Xgraphics
2	erase all objects
3	erase background/Xgraphics and objects
4	erase leftover objects
5	erase background/Xgraphics and leftover objects
10	same as 2, except only for use immediately beforeRec(6)
11	same as 3, except only for use immediately before Rec(6)
33	same as 1, except also erase AutoRedraw image
35	same as 3, except also erase AutoRedraw image
65	same as 1, except also erase AutoRedraw and BgndPicture images
67	same as 3, except also erase AutoRedraw and BgndPicture images

To erase only those objects with IDNumbers within a certain range, express the range via the FromID# and ToID# parameters. Refer to the EraseFromID and EraseToID attributes for details.

For additional information about the various settings, see EraseType.

Notes For proper operation, include a space between ERASE and (.

Also see Condition, Destroy, EraseType

Erase Object

Description — Removes objects and/or Xgraphics from a window.

Attributes
Comment
Condition
DoEvents
EraseFromID
EraseToID
EraseType
Name

Details — The Erase object is often used at or near the top of a page to erase what was previously added to a window in preparation for the current page.

Also see — AutoRedraw, Condition, Destroy, ERASE, EraseFromID, EraseType

EraseFromID Attribute

Applies to	Erase object
Description	Controls the start of the range of objects that are erased from the window.
Settings	0 to 99
Details	EraseFromID, coupled with EraseToID, let you control the range of the objects that are erased from the window. Objects with IDNumbers within the range you specify are removed from the window. Remember that IDNumbers are arbitrary numeric values you assign each object, and that two objects of different classes can have the same IDNumber. For example, a window could contain a Textbox with IDNumber 1 as well as a Picture with IDNumber 1. If the value 1 falls within the EraseFromID and EraseToID range, both the Textbox and the Picture will be erased. To erase all objects, set both EraseFromID and EraseToID to 0.
Example	By selecting IDNumbers carefully, you can arrange to remove only certain ones from the window with the Erase object. If you have a button bar at the top of the window, you probably want the bar to remain in the window without replotting as the user moves from page to page. You can assign the buttons IDNumbers from 90 to 99, and set the EraseFromID to 0 and EraseToID to 89. That will erase all objects except those of the button bar.
Notes	The EraseType must be set to a value greater than 1 for objects to be removed.
Also see	Destroy, EraseType, IDNumber, Obj() Function

EraseToID Attribute

Applies to	Erase object
Description	Controls the end of the range of objects that are erased from the window.
Settings	0 to 99
Details	Refer to the EraseFromID attribute.

EraseType Attribute

Applies to Erase object

Description Specifies the items to be removed from the window.

Settings

0	do nothing
1	erase drawn graphics/text
2	erase objects in window
3	erase drawn graphics/text and objects
4	erase leftover objects
5	erase draw graphics and leftover objects
16	copy image of window into background (experimental)
18	same as 16, but then erase objects (experimental)
33	same as 1, except also erase AutoRedraw image
35	same as 3, except also erase AutoRedraw image
65	same as 1, except also erase AutoRedraw and BgndPicture images
67	same as 3, except also erase AutoRedraw and BgndPicture images

Details Most authors place an Erase object at the top of a page, and set EraseType to 3; this removes objects (such as Textboxes, Buttons, etc.) as well as items drawn onto the window itself, as described below.

INFLUENCE OF AUTOREDRAW

The AutoRedraw setting of the window influences what EraseType settings 1 and 3 do. If AutoRedraw is enabled (such as by a prior Layout object) when the Erase object is encountered at run time, then settings 1 and 3 remove everything drawn on the window itself, including text displayed via DrawText,backgrounds created with Tile, and graphics drawn via Xgraphics Commands.

If AutoRedraw is not enabled when the Erase object is encountered at run time, then EraseType settings 1 and 3 do *not* remove those items (if any) drawn onto the window itself during the period AutoRedraw had been enabled. Note that DrawText setting -2 and Tile backgrounds automatically, temporarily enable AutoRedraw.

FORCING ERASURES

If AutoRedraw is not enabled when the Erase object executes, but you still you want to remove drawings created when AutoRedraw was enabled, then add 32 to the EraseType setting (i.e. use EraseType setting 33 or 35).

Alternatively, enable AutoRedraw in the Layout object.

ERASING BGNDPICTURE

Images displayed in the window via the Layouts BgndPicture attribute (when Tile is disabled) are not normally removed by the Erase object. If you want to erase such an image, set EraseType to 65 or 67.

Notes For smoother and faster page-to-page transitions, try EraseType 4 in an Erase object

placed immediately before a Wait object in the page. EraseType 4 removes objects that do not appear in the current page between its start and this Erase object; such objects are usually left over from a prior page. Note that objects added via an Include object are not considered to be in the page.

EraseType 16 and 18 are experimental features (i.e. use at your own risk). Authors use EraseType 18 to provide a more interesting visual transition to the next page. EraseType 18 takes a "photograph" of the contents of the window (including objects), transfers that photo into the BgndPicture of the window, then erases the objects normally (i.e. as if EraseType 2 was in use). The objects are removed, but their image remains. Typically, authors immediately follow such an Erase object with a Layout object that loads a different BgndPicture with a SpecialEffect. This technique creates a very smooth visual transition to the next page.

An EraseType setting of 0 performs DoEvents normally.

Also see AutoRedraw, ERASE, EraseFromID, LockUpdate

EventVar Attribute

Applies to	Wait object
Description	Specifies the name of the variable in which to store the most recently received event code.
Details	Only those event codes that match one of the Wait object's activators (NextActivator, BackActivator, etc.) are stored in the EventVar. Exception: if you enter anything for the AllOtherAction attribute, Everest puts all event codes in the EventVar.
Also see	Sysvar(12)

Execute Attribute

Applies to	OLE object
Description	Enter a string for the server application to execute.
Details	Set the Action attribute to 8 to execute the string. Protocol must be set to "StdExecute" for this to work. Check the server application's documentation to determine what Execute strings it supports.
Also see	Action

Ext() Function

Applies to A-pex3 programming

Description The Ext() function serves as the "catch all" for unusual functions. Many are carry-overs from Summit for DOS.

Syntax ext(Operation [, Items])

Operation is a number that specifies the action to perform.

Items are additional parameters; not all Operations have Items.

Details The value returned by the Ext() function depends on the Operation code. Those described as "Reserved" are functions supported in Summit for DOS, but not in Everest. Some Operations accept multiple parameters:

Usage	Returns
ext(-29)	the disk path to the currently executing program
ext(-28)	the disk path to the Windows System directory
ext(-27)	the disk path to the Windows subdirectory
ext(-26)	the default disk path on the Z: drive
.	
.	
ext(-1)	the default disk path on the A: drive
ext(0)	the current DOS default disk path
ext(1, X, Y [, Window])	the color of the pixel at location X, Y in the current window; include the optional Window parameter to read from a different Everest window
ext(2)	the microprocessor type
ext(3, Segment, Offset)	peeks at the byte at memory location Segment:Offset; Windows does not handle this reliably; use at your own risk
ext(4, Segment, Offset, Numeric)	pokes Numeric to memory location Segment:Offset; Windows does not handle this reliably; use at your own risk
ext(5)	non-zero if an event (key press or mouse button click) is pending in the Windows event queue
ext(6)	the number of musical notes still waiting to be played via the Ply() Function

ext(7 to 18)	Reserved; Summit for DOS users should see the Fyl() Function for a substitute.
ext(19)	sends a copy of the image in the current window to the printer
ext(20)	Reserved
ext(21)	Reserved
ext(22)	if an EGA display adapter is installed, this Action returns the amount of display memory (in K); VGA always returns 256
ext(23)	the version of Everest; also see ext(123)
ext(24)	Reserved
ext(25)	the number of visible text columns in the current window for text PRINTed in the current font; uses the character "0" as reference for average character width
ext(26)	the number of visible text lines in the current window for text PRINTed in the current font; uses the character "0" as reference for average character height
ext(27)	disables normal event handling, polls for the next event (keypress, mouse click, etc.), and returns the numeric code of that event, 0 if none were waiting in the event queue; returns different keypress codes than Summit for DOS; IMPORTANT: this function disables normal event handling until you employ the key(0) function to return event control to Windows; WARNING: disabled event handling may be confusing to the user, and may make your project difficult to debug; use at your own risk since once you disable event handling the only way to enable it again is via key(0)
ext(28)	disables normal event handling, waits for the next event (keypress, mouse click, etc.), and returns the numeric code of that event; returns different keypress codes than Summit for DOS; IMPORTANT: this function disables normal event handling until you employ the key(0) function to return event control to Windows; WARNING: disabled event handling may be confusing to the user, and may make your project difficult to debug; use at your own risk since once you disable event handling the only way to enable it again is via key(0)
ext(29)	computer performance indicator; a number related to the speed of the microprocessor (the larger the number, the faster the computer)
ext(30 to 35)	Reserved.
ext(36)	the number of author defined variables in use

ext(37 to 38) Reserved.

ext(39) returns a number that indicates the mode in which the project is running: -1 = authoring, 0 = debugging, 1 = user

ext(40) Reserved.

ext(41, FilePattern)
the name of the first file on disk that matches FilePattern; for example, if FilePattern is "C:*.bat" the function might return a file name such as AUTOEXEC.BAT

ext(42) the name of the next file on disk that matches the FilePattern used in the most recent ext(41) function call; when null, no more files match

ext(43) the command line parameters specified when the program was started

ext(44) makes a hidden mouse cursor visible again

ext(45) hides the mouse cursor (temporarily)

ext(46 to 48) Reserved.

ext(50) a non-zero number if a color monitor is in use, 0 for monochrome

ext(51) call interrupt; before referencing ext(51), store interrupt number in Sysvar(120), AX in Sysvar(121), BX in Sysvar(122), CX in Sysvar(123), DX in Sysvar(124); results are returned in the same Sysvars; only AX through DX are supported; intended for advanced programmers only

ext(101) yields to allow Windows to process its event queue; returns the number of open windows; due to a bug in Windows, do not use in a ChangeEvent or in a Program object that has Refresh set to Yes; also see DoEvents attribute

ext(102, PortNumber)
printer status bits for parallel port PortNumber; PortNumber ranges from 1 to 4; returns 0 if the printer is ready and waiting for data; other values can be decoded by examining the bits: bit 7 = busy, 6 = acknowledge, 5 = out of paper, 4 = selected, 3 = I/O error, 2 = unused PortNumber, 1 = no printer available, 0 = time out. Employ the ^? operator to test a bit.

ext(103, WindowHandle)
calls the Windows API LockWindowUpdate function. For example, to temporarily disable plotting of objects in the current window use: `dummyvar = ext(103, Window(0).hwnd).` When ready to let pending objects plot (all at once), use `dummyvar = ext(103, 0).` Be sure to eventually unlock a locked window. Use at your own risk. Also see LockUpdate.

ext(104, ChildWindowHandle, NewParentWindowHandle)
calls the Windows API SetParent function. Use at your own risk.

ext(105) sums Level 1 scoring values in Sysvar(5) and Sysvar(6) into Level 2 scoring in Sysvar(105) and Sysvar(106); resets Sysvar(5) and Sysvar(6) to 0; returns the value in Sysvar(107), the current Level 2 score

ext(106, hwnd, uMsg, wParam, lParam)
calls the Windows API SendMessage function. Use at your own risk.

ext(107) retrieves the text (if any) in the Windows clipboard

ext(108, Text) places Text in the Windows clipboard

ext(109, Text) returns the display length (in pixels) of Text as it would be shown in the window via the PRINT command; for Picture objects, set sysvar(108) to the IDNumber of the desired object before calling this function, then reset sysvar(108) back to 0

ext(110, Err) causes error number Err to occur (for debugging purposes)

ext(111) returns the horizontal AutoResize scaling factor for the current window

ext(112) returns the vertical AutoResize scaling factor for the current window

ext(113, RGBColor, OldGradations, NewGradations)
scales an RGB color value from one range of possible color gradations to a new range of gradations, and returns the new RGB value; for example, if the intensity of RGBColor has red, green and blue components that each are in the range of 0 to 255, and you want each to range from 0 to 7, you would use `newrgb=ext(113, oldrgb, 256, 8).`

ext(114) returns the number of installed devices capable of playing .WAV audio

ext(115, String) constructs and returns a seemingly "random" password consisting of 9 digits that is based on the current year and the String you specify; can be used in your project to restrict end-user access; returns the same number for the entire current year; see usage example below

ext(116, String) same as ext(115), except returns a number that is the same for the current month

ext(117, String) same as ext(115), except return a number that is the same for the current day

ext(120) the number of files in the Internet/intranet download cache

ext(121) deletes all files in the Internet/intranet download cache

ext(122, Port) returns the status of the specified serial communications port as a 16-bit number; the bits are as follows: 15=carrier detect, 14=ring indicator,

13=data set ready, 12=clear to send, 11=CD changed, 10=RI changed, 9=DSR changed, 8=CTS changed, 7=timeout, 6=transmitter empty, 5=transmit holding register empty, 4=break detected, 3=framing error, 2=parity error, 1=overrun error, 0=data ready

ext(123) the file date and time of the currently executing Everest .EXE; convert to date format via `fmt(ext(123), General Date)`; also see ext(23)

ext(124) the number of windows currently open in your project

ext(125, String)
calculates a mathematical expression in String and returns the result; Example: `ext(125, "1+2")` returns 3.

ext(126, Visible)
opens or closes the INet History window; if Visible is 1 the window opens; if Visible is 0, the window closes.

ext(130) the DOS version

ext(131) the Windows version (returns 3.95 for Windows 95)

Example The following A-pex3 example demonstrates how your project could limit access to only those users with today's password:

```
realpass = ext(117, "any_secret")   $$ today's password
userpass = ibx("Please enter the password for " + dat(0))
IF userpass # realpass THEN         $$ wrong password!
  dummyvar = mbx("Call us to obtain today's password.", 16)
  BRANCH @exit                      $$ exit project
ENDIF
```

To determine today's password yourself (in order to supply the correct one to the end user), in the AUTHOR program, from the Utilities pull-down menu, choose Evaluate Expression. Then enter `ext(117, "yourword")`, where `yourword` is the secret string you used in your project. Note that this feature requires that the current date on your computer as well as the end user's is the same.

Also see Sysvar() Variables

FadeIn Attribute

Applies to	Animate object
Description	Controls the transition effect before the start of the animation.
Settings	0 no fade 1 fade from black 2 fade from white
Notes	Due to a bug in Autodesk's animation player, a non-zero FadeIn value can cause the animation image to become invisible unexpectedly. Use at your own risk.
Also see	FadeOut

FadeOut Attribute

Applies to	Animate object
Description	Controls the transition effect at the end of the animation.
Settings	0 no fade 1 fade to black 2 fade to white
Notes	Due to a bug in Autodesk's animation player, a non-zero FadeOut value can cause the animation image to become invisible unexpectedly. Use at your own risk.
Also see	FadeIn

FBOX Command

Applies to A-pex3 Xgraphics programming

Description Draws a filled rectangle.

Syntax FBOX (X1, Y1, X2, Y2 [, Color])

Details The FBOX command draws a solid rectangle in the window. Specify the coordinates of two opposite corners in pixels via the X1, Y1, X2 and Y2 parameters. Include the Color parameter only if you want the box to be drawn using one of the 16 palette colors; otherwise Everest uses the current foreground color. You can set the current foreground color via the COLOR command.

FBOX ignores the current STYLE setting for painting and paint color; FBOX always fills the inside of the box.

Example The following example draws a filled bright green box:

```
COLOR (-1, 0, 255, 0)
FBOX (50, 50, 100, 100)
```

Notes For proper operation, include a space between FBOX and (.

Also see BOX, GFILL, PAINT, STYLE

FileName Attribute

Applies to	Media object
Description	Specifies the name of the disk file that contains the media object to play or record.
Details	Before choosing a file name, or clicking on the "more information" arrow, set the DeviceType. Certain DeviceTypes (such as CDAudio) do not use a FileName. Instead, control the playback for these DeviceTypes via the EndAt and StartAt attributes. For help with file locations, refer to Appendix F.
Example	To play a .WAV sound file, set FileName to the name of the desired .WAV file, set DeviceType to WaveAudio and set Command to Play.
Also see	DeviceType, EndAt, StartAt

FillBarColor Attribute

Applies to	Gauge object
Description	Determines the color used for the bar that fills a bar-style gauge.
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.
Also see	GaugeStyle

FillColor Attribute

Applies to	Button, Check, Combo, Frame, Gauge, Listbox, Option, Shape objects
Description	Sets the color used to fill the area of the object; can be thought of as the background color of the text caption (for those objects with text).
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.
Also see	CaptionColor

FillStyle Attribute

Applies to Shape object

Description Sets the type of pattern used to fill in the inside of a shape.

Settings

0	solid
1	transparent (no fill)
2	horizontal lines
3	vertical lines
4	NW SE diagonal lines
5	SW NE diagonal lines
6	crossing lines
7	diagonal crossing lines

Example Shapes appear on top of the BgndPicture. To easily detect a user click on a part of a picture, load the image via the Layout's BgndPicture attribute, then put Shape(s) on top. Set the FillStyle of the Shapes to 1 (transparent) so only their outlines are visible. Be sure to set the ClickEvent to the desired value.

Also see FillColor, STYLE

FillValue Attribute

Applies to Gauge object

Description Sets the value to be represented by the Gauge.

Details The FillValue must be within the range specified by the Min and Max attributes.

The FillValue you set in the Attributes window becomes the initial value of the Gauge. Use A-pex3 programming code to alter the FillValue at project run time.

Examples If Min is set to 0, Max is 100, and FillValue is 75, the needle or bar of the Gauge will display a "3/4 full" value.

The following A-pex3 programming example sets the style of the Gauge with IDNumber 1 to the 360 degree needle type, and makes the needle move like the second hand on a watch. To run this example, place a Gauge object in your page followed by a Program object that contains the following code:

```
Gauge(1).GaugeStyle = 3       $$ needle style
Gauge(1).Min = 0
Gauge(1).Max = 59
DO
  tim3 = tim(3)               $$ seconds (of current time)
  Gauge(1).FillValue = tim3
  dummyvar = ext(101)         $$ allow gauge refresh
  DO IF tim(3) = tim3: LOOP   $$ wait until next second
LOOP IF ext(5) = 0      $$ loop until next event
```

Also see GaugeStyle, Max, Min

FindString Attribute

Applies to Combo, Listbox objects

Description Searches for an item that matches the specified text string, and returns the result in FoundIndex.

Details To perform a search, set FindString equal to the text for which you want to search. FindString looks for a match among the items starting with the item after that pointed to via FoundIndex. The search is performed on a case-insensitive basis.

After setting FindString, check the value of FoundIndex. A FoundIndex value of 0 or more indicates the location at which the FindString was found. If the FindString was not found, FoundIndex returns -1.

Note that while searching, FindString can wrap back to the first item. Therefore, to exit a DO...LOOP that searches for all matching items, compare FoundIndex to its previous value, and if it is less, use the OUTLOOP command.

Example The following A-pex3 programming example searches the Listbox with IDNumber 1 for all items that equal Maryland, and highlights them:

```
Listbox(1).FoundIndex = -1    $$ start at the top
lastfound = 0
DO
  Listbox(1).FindString = "Maryland"
  IF Listbox(1).FoundIndex < lastfound THEN OUTLOOP
  lastfound = Listbox(1).FoundIndex
  Listbox(1).LookAt = lastfound
  Listbox(1).Tagged = -1
LOOP
```

Also see FoundIndex

Flextext Object

Description — The Flextext object displays one or more lines of text to the user. Flextext can display text in multiple colors and sizes. Optionally, a Flextext object can contain hypertext jump and popup words that generate an event when the user clicks on them. For a description of how to employ these special formatting and hypertext features, refer to the Text attribute.

Attributes

AnimPath
BackColor
Bottom
ClickEventt
Comment
Condition
Create
DblClickEvent
Destroy
DragMode
Enabled
FocusRect
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
ForeColor
HeadingSize
Height
IDNumber
Initially
JumpPointer
Left
MouseLeaveEvent
MouseOverEvent
Move
NormalPointer
PopupPointer
Right
SaveAsObject
TabOrder
TabStop
Text
Top
Update
Visible
Width
Zev
ZOrder

Details — Use a Flextext object when you have several lines of text to display using more than one

foreground color or size, or when you need hypertext features. Contrast this with the simpler Textbox object that displays text in one color.

If the text does not fit inside the box, the user can scroll it up/down. A Flextext object can contain up to 32,000 characters.

Flextext text plots in a destructive fashion (i.e. it erases whatever it plots on top of). To display non-destructive (sometimes called transparent) text, use the A-pex3 PRINT and FONT commands, or a Textbox with DrawText enabled.

If you need a Textbox that is editable by the user, employ an Input Object.

Notes — There is a known bug in the Flextext object that makes objects (such as Buttons) placed on top of the Flextext object require TWO mouse clicks to focus on/activate. This problem has been reported to the Flextext object's developer.

Also see — Input Object, PRINT, Textbox Object

Fmt() Function

Applies to A-pex3 programming

Description A very powerful function that converts numbers and strings into specialized forms that are particularly useful for data display.

Syntax fmt(Value, Style)

Value is a number or character string to be formatted.

Style is a character string that indicates the type of formatting you want.

Details Style can be one of several predefined formatting styles, or a string of characters that specifies a user-defined format.

PREDEFINED STYLES

When Value is a number, employ any of the following predefined Styles to have the function return the Value in the indicated form:

Currency	in currency form
Fixed	at least one digit on left, and two on right of decimal separator
General Number	as a number (no special formatting)
On/Off	"Off" if value is 0, otherwise "On"
Percent	multiply by 100, append %, and two digits to the right of the decimal separator
Scientific	scientific notation
Standard	with thousands separator and two digits to the right of the decimal separator
True/False	"False" if value is 0, otherwise "True"
Yes/No	"No" if value is 0, otherwise "Yes"

Example The following example converts the number 12345 into Scientific notation and stores the result in the variable named sci:

```
sci = fmt(12345, "Scientific")
```

USER-DEFINED STYLES

Style can also be user-defined. Typically, the Style string is a series of formatting characters that apply to the Value, digit by digit. Employ the following characters in the Style string to format the number as indicated:

""	display the number with no formatting
0	digit or 0; if there is a digit in Value in the character position of the 0, return the digit, otherwise return 0 in that position
#	digit or nothing; if there is a digit in Value in the character position of the #, return the digit, otherwise return nothing in that position
.	decimal placeholder; marks the location of the decimal separator; the actual character used as the decimal separator is defined by the Number Format specified in the International section of the Windows Control Panel
%	percentage conversion; Value is multiplied by 100 and a % is included
-+$()Space	literal; display the character indicated
\	force literal; display the next character as is
*	fill character; employ the character that follows * to fill in empty areas

Example The following example stores "12345.00" in the num variable:

```
num = fmt(12345, "00000.00")
```

MULTIPLE SECTIONS

User-define Styles can have up to three sections; each section applies to Values within a certain range. Sections are separated with a semicolon. When you specify one section, it applies to all Values. With two sections, the first applies to Values greater than or equal to 0, and the second to Values less than 0. With three sections, the first applies to Values greater than 0, the second to Values less than 0, and the third to 0.

Example The following example stores "zilch" in the num variable:

```
num = fmt(0, "00000.00;00000.00;\z\i\l\c\h")
```

HANDLING DATES AND TIMES

The Fmt() function can also handle date and time information. Employ the following predefined Styles:

General Date	date in form 3/11/95
Long Date	date in long form specified by the International section of the Windows Control Panel

Medium Date	date in short form, except spell out month abbreviation
Short Date	date in short form
Long Time	time in long form specified by the International section of the Windows Control Panel
Medium Time	time in 12-hour form
Short Time	time in 24-hour form

Examples The following example stores the current date with month abbreviation in the variable named nowdate:

```
nowdate = fmt(dat(""), "Medium Date")
```

The following example stores the current time in 12-hour form in the variable named nowtime:

```
nowtime = fmt(tim(""), "Medium Time")
```

USER-DEFINED DATE STYLES

Employ the following characters in Style to create your own date formats. Be sure to pass a date string in Value.

d	day of month as a number without leading zero (1 to 31)
dd	day of month as a number with leading zero (01 to 31)
ddd	day of week name as an abbreviation (Sun to Sat)
dddd	day of week name (Sunday to Saturday)
ddddd	date in Short Date form
dddddd	date in Long Date form
m	month as number without leading zero (1 to 12)
mm	month as number with leading zero (01 to 12)
mmm	month name as an abbreviation (Jan to Dec)
mmmm	month name (January to December)
q	quarter of the year as a number (1 to 4)
y	day of year as a number (1 to 366)
yy	year as a two digit number (00 to 99)

yyyy	year as a four digit number (1000 to 9999)

Examples The following example stores a day of the year number in the variable named daynum:

```
daynum = fmt("03/11/95", "y")
```

The following example stores "March 11, 1995" in the variable named textdate:

```
textdate = fmt("03/11/95", "mmmm dd\, yyyy")
```

USER-DEFINED TIME STYLES

Employ the following characters in Style to create your own time formats. Be sure to pass a time string in Value.

AM/PM	12-hour form with AM or PM
am/pm	12-hour form with am or pm
h	hour as a number without leading zero (0 to 23)
hh	hour as a number with leading zero (00 to 23)
n	minute as a number without leading zero (0 to 59)
nn	minute as a number with leading zero (00 to 59)
s	second as a number without leading zero (0 to 59)
ss	second as a number with leading zero (00 to 59)

Example The following example stores "09:35 PM" in the variable named bigtime:

```
bigtime = fmt("21:35", "hh:mm AM/PM")
```

USER-DEFINED STRING STYLES

Typically, the Style string is a series of formatting characters that apply to the Value, character by character. Employ the following characters in the Style string to format Value as indicated:

@	character or space; if there is a character in Value in the position of the @, return the character, otherwise return a space in that position
&	character or nothing; if there is a character in Value in the position of the &, return the character, otherwise return nothing in that position
<	convert to lower case

>	convert to upper case
!	left justify

Example The following examples both store the word Everest in the variable named sysname right justified within a 10-character area:

```
sysname = fmt("Everest", "@@@@@@@@@@")

sysname = fmt("Everest", "@" $ 10)
```

Also see Dat() Function, Tim() Function

Fnt() Function

Applies to A-pex3 programming

Description Returns information about fonts installed in Windows.

Syntax fnt(Operation, Device [, Find])

Details The operation of the Fnt() function depends on the value of the Operation parameter:

Operation	Result
0	returns the number of fonts installed on the Device.
> 0	returns the name of a specific font installed.
-1	searches for a font with the name Find and returns a number greater than 0 if found, or 0 if not found.

Device can be either 0 (display font) or 1 (printer font).

Example The following A-pex3 program determines if a Courier font is available to the Windows Print Manager:

```
IF fnt(-1, 1, "Courier") = 0 THEN
  dummyvar = mbx("Courier font not available.")
ENDIF
```

Also see FONT, FontName

Focus Attribute

Applies to	OLE object
Description	Determines whether Everest attempts to transfer the focus to the server application upon Action 7 (activate).
Settings	Yes attempt to transfer focus No do not attempt to transfer focus
Also see	Action

FocusRect Attribute

Applies to	Flextext object
Description	Determines whether a dashed line appears around the object at run time when it has the focus.
Settings	Yes show dashed line when object has the focus No do not show dashed line when object has the focus
Also see	BorderStyle

FONT Command

Applies to A-pex3 Xgraphics programming

Description Sets various attributes of text subsequently printed via the PRINT command.

Syntax FONT ([FontName], [Size], [Transparency], [Bold], [Italic], [StrikeThru], [Underline])

Details Use the FONT command to select a typeface, as well as choose its attributes. Place the FONT command before the PRINT command(s) you want to influence.

You need not include all parameters. If you leave out a parameter, but include others that follow, include a comma for each skipped parameter. Everest retains the previous value for parameters that you omit.

FontName	the name of the Windows font; to obtain a list of names of fonts on your computer, invoke the font selection dialog box for an object such as a textbox (see FontName Attribute); if you employ a font that does not come with Microsoft Windows, you should license it and include it when you ship your project
Size	the size of the font in points (72 points = 1 inch)
Transparency	0 = not transparent (erase Xgraphics underneath) -1 = transparent (merge with what is underneath)
Bold	0 = not bold -1 = bold
Italic	0 = not italic -1 = italic
StrikeThru	0 = do not draw line through text -1 = draw line through text
Underline	0 = do not underline text -1 = underline text

Example

```
FONT ("Roman", 17,, 0)
PRINT (100, 100,, "This is Roman font.")
FONT (,,, -1)
PRINT (100, 150,, "This is Roman font bold.")
```

Notes The font selections you make remain with the window until changed (i.e. the settings do not automatically revert back to the default values).

For proper operation, include a space between FONT and (.

Also see PRINT

Font3d Attribute

Applies to	Button, Check, Combo, Frame, Listbox, Option objects
Description	Controls the 3-dimensional appearance of the text printed on the object.
Settings	0 no 3-D effect 1 raised 2 raised with more shading 3 lowered 4 lowered with more shading
Details	The appearance of the text is also influenced by the colors you choose. In fact, certain color combinations can make the 3-D effect appear reversed (i.e. raised text becomes lowered). Experiment to find the effect you want.
Also see	CaptionColor

FontBold Attribute

Applies to	Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Option, Textbox objects
Description	Determines whether the font used to display text is bolded.
Double click	Opens font dialog box. You can visually choose font name, size and style.
Settings	Yes use bold font No use non-bold font

FontItalic Attribute

Applies to	Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Option, Textbox objects	
Description	Determines whether the font used to display text is italicized.	
Double click	Opens font dialog box. You can visually choose font name, size and style.	
Settings	Yes	use italicized font
	No	use non-italicized font

FontName Attribute

Applies to — Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Option, Textbox objects

Description — Determines the font used to display text.

Double click — Opens font dialog box. You can visually choose font name, size and style.

Details — Everest lets you employ any Windows font (including bitmapped, vector and TrueType). Click on the "more information" arrow of the Attributes window to select from a list of the fonts available on your system.

Note that FontName is NOT the name of a disk file that contains the font. Instead, it is a name that Windows associates with the font. Therefore, you should not prefix a FontName with a disk path.

In general, you should select the FontName before setting other font attributes.

To ensure that the user's computer has the font you employ, you should use only those supplied with Windows itself. Alternatively, you can create your own fonts with a package such as Fontographer (by Altsys), and supply them with your project. The following fonts are included with Windows 3.1:

Arial
Courier
Modern
MS Sans Serif
MS Serif
Roman
Script
Symbol
Times New Roman
Wingdings

If the user's computer does not have the font you employ, Windows will pick another one it believes to be most similar. Usually the results are not very attractive. You can employ the Fnt() Function to determine at run time if a particular font is available on the user's computer.

Also see — Fnt() Function, FontSize

FontSize Attribute

Applies to — Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Option, Textbox objects

Description — Controls the size of the text printed within the object.

Double click — Opens font dialog box. You can visually choose font name, size and style.

Details — Because not all sizes are available for all fonts, it is best to set the FontSize via the "more information" arrow of the Attributes window.

Set FontName before you set FontSize.

Windows does not accurately scale fonts to match the display resolution. Consequently, the size of text relative to its container (such as a Button) differs. For example, when Windows runs in 1024 x 768 resolution, text can appear larger than in 640 x 480 resolution. We recommend you try running your project in different Windows screen resolutions to learn how it will appear to various users.

To compensate for this weakness in Windows, you might consider enabling Everest's feature that scales fonts at run time. To do so, set Sysvar(115) to the desired scaling factor. For example:

```
IF sysvar(3) = 12 THEN  $$ if fonts too large
  sysvar(115) = 12/15   $$ make them 80% as big
ENDIF
```

Sysvar(115) works only on text that has a FontSize attribute. Note that scaled fonts may not look as attractive as non-scaled fonts (another limitation of Windows).

Also see — FontName, Sysvar(115)

FontStrikeThru Attribute

Applies to	Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Option, Textbox objects
Description	Determines whether a line is drawn through text.
Settings	Yes draw line through text No do not draw line through text
Details	Accessible via A-pex3 programming only.
Example	`Textbox(1).FontStrikeThru = "Yes"`

FontUnderline Attribute

Applies to	Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Option, Textbox objects
Description	Determines whether a line is drawn under text.
Settings	Yes underline text No do not underline text
Details	Accessible via A-pex3 programming only.
Example	`Textbox(1).FontUnderline = "Yes"`

ForeColor Attribute

Applies to	Flextext, Input, Layout, Mask, Textbox objects
Description	Controls the color of the text or caption displayed within the object.
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.

Format Attribute

Applies to Mask, OLE objects

Description For a Mask object, determines the display formatting applied to the contents. For an OLE object, determines the format of data exchanged with the server.

Details **MASK OBJECT**

The Format attribute expresses the formating you want the Mask object to apply to the user's input. For example, by setting Format to 0%, after entry the user's input will automatically be converted to a percent form (i.e. multiplied by 100 and appended with a % sign).

You can enter the same expressions as you do for the fmt() function Fmt() Function, with the exception that named formats (such as "Currency") cannot be used.

OLE OBJECT

After setting the other OLE object attributes, double click the Format attribute in the Attributes window to view a list of Formats supported.

Also see Fmt() Function, InputTemplate

FoundIndex Attribute

Applies to	Combo, Listbox objects
Description	Sets the starting location of the next FindString text search, and returns the result of a FindString text search.
Details	Before using FindString, set FoundIndex to the number of the item at which to begin searching, minus 1. For example, to start with the first item (item number 0), set FoundIndex to -1. Later, after setting FindString, examine the value in FoundIndex to determine if and where the text was found. A value of -1 means the FindString text was not found. Any other value is the number of the item that matches FindString.
Example	Refer to the example for FindString.
Also see	FindString

Frame Object

Description The Frame object displays a rectangular area in the window, and is often used to visually group other objects. The Frame has the unique ability to rotate its text caption.

Attributes
Alignment
BorderColor
BorderStyle
Botton
Caption
CaptionColor
CaptionRotation
ClickEvent
Comment
Condition
Create
Destroy
DragMode
EdgeDistInside
EdgeSize
EdgeStyle
EdgeStyleInside
Enabled
FillColor
Font3d
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
Height
IDNumber
Initially
Left
LetterRotation
LightColor
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
MultiLine
Name
Pic
Right
SaveAsObject
ShadowColor
Top
VerticalAlignment
Visible
Wallpaper

Width
ZOrder

Details

By manipulating the edge attributes for size and color, you can obtain attractive 3-D effects.

At design time, if you place other objects, such as Check boxes on a Frame, then edit the Frame, the other objects will be temporarily obscured by the Frame. To reveal such hidden objects, first focus on the Frame (i.e. click on it to bring the sizing handles to it) then choose the "Push to Z-back" option on the main Edit pull-down menu.

Most authors set the ZOrder of Frame objects to 1 so that they appear behind other objects at run time.

Fre() Function

Applies to A-pex3 programming

Description Returns the amount of unused memory available to Windows, or the amount of unused disk space.

Syntax

fre(Numeric)	to check memory or resources
or	
fre(String)	to check disk space

Details Use this to monitor the amount of memory and/or disk space available to your project.

Example	Information Returned
fre(-1)	total amount of free memory available, in bytes
fre(0)	% of system resources still available
fre(1)	% of GDI resources still available
fre(2)	% of USER resources still available
fre("A:")	unused disk space on drive A:, in bytes

In general, Windows resources are consumed whenever you display something in a window. Certain object classes, especially Picture, consume a fairly significant amount of resources.

When available resources drop to about 20%, Windows performs more sluggishly, may fail to properly display or refresh windows, and could hang.

By monitoring memory and resources with the Fre() function, you can estimate when Windows is unable to run your project smoothly, and exit gracefully.

Examples The following example checks available memory and resources, and exits if it deems the values too low:

```
IF fre(-1) < 400000 @ fre(0) < 20 THEN
  message = "Not enough available memory!"
  dummyvar = mbx(message, 0)
  BRANCH @end
ENDIF
```

The following A-pex3 program checks if there is enough disk space before it copies a file:

```
fromfile = "C:\project\picture.pcx"
tofile = "D:\backup\picture.pcx"
message = ""
IF fre(tofile) >= fyl(-2, fromfile) THEN
  ecode = fyl(-5, 1, fromfile, tofile)
```

```
  IF ecode # - 1 THEN message = "Error during copying"
ELSE              $$ ok to copy via fyl(-5...)
  message = "Not enough space on drive " + (tofile\2)
ENDIF
IF len(message) > 0 THEN dummyvar = mbx(message, 0)
```

Notes — If an error occurs while checking the amount of disk space, Fre() returns -1 and puts the error code in Sysvar(1) .

Also see — Ext() Function, Fyl() Function

Functions

Applies to A-pex3 programming

Description A function accepts one or more parameters enclosed by parentheses, and returns a value. In Everest, all functions have three letter names.

Details Everest offers many functions for use in your programs. Functions can be used anywhere a variable is allowed. For example, the Len() function returns the number of characters in a string.

```
slength = len("Everest")
```

stores the value 7 in the variable named slength because there are seven characters in the word Everest.

Because functions always return information, in programming, you must specify a variable to hold the returned information, even if it will be ignored. For example:

```
ext(19)                 $$ improper syntax!

dummyvar = ext(19)      $$ correct syntax
```

The parameters of functions can be constants, variables, or even functions. Functions can be nested up to 8 levels deep. Multiple parameters are separated by commas.

Functions are documented individually in this technical reference, and can be found alphabetically. Below is a list of them grouped according to functionality:

NUMERIC

Abs()	absolute value
Atn()	arctangent
Cos()	cosine
Hex()	hexadecimal
Log()	natural logarithm
Lwr()	round down to nearest integer
Rgb()	combines red, green, blue
Rnd()	random number
Sel()	unique random number
Sin()	sine
Sqr()	square root
Tan()	tangent
Upr()	round to nearest integer

STRING (for string parsing see Operators)

Asc()	ASCII value
Chr()	character
Fmt()	formatting for output
Len()	string length

Ltr()	remove leading blanks
Lwr()	convert to lower case
Mid()	copy a portion of a string
Pik()	return one item from a list
Rpl()	replace or remove a character
Rtr()	remove trailing blanks
Upr()	convert to upper case
Val()	convert to numeric

VARIABLES AND ARRAYS

Arr()	number of elements in an array
Lod()	copy values in an array
Srt()	sort an array
Sum()	sum elements of an array
Typ()	storage form of contents of variable
Var()	test if variable exists

DISK/OPERATING SYSTEM

Env()	environment
Fnt()	determine available fonts
Fre()	free memory and disk space
Fyl()	read/write disk files
Gdc()	Windows GetDeviceCaps
Gsm()	Windows GetSystemMetrics
Hlp()	Windows Help system
Ini()	read/write .INI files
Msg()	read EVEREST.MSG file
Pth()	parse file names
Rec()	read/write records files
Scn()	page exists

EXTERNAL DEVICES AND APPLICATIONS

Dde()	Windows Dynamic Data Exchange
Dll()	call Dynamic Link Library routine
Mci()	Windows Media Control Interface
Mse()	mouse
Ply()	play musical notes
Shl()	execute external application

MISCELLANEOUS

Bbt()	calls the Windows API BitBlt function
Cvi()	convert Mki() string to integer
Dat()	current date
Ext()	assorted
Ibx()	user input box
Key()	keyboard
Mbx()	message box

Mki()	convert integer to 2-byte string
Obj()	Everest objects
Reg()	return region string
Sfl()	shuffle attributes of a group of objects
Tim()	current time
Wrp()	wrap text

Also see Attributes, Commands, Operators, Program Object

Fyl() Function

Applies to A-pex3 programming

Description Accesses (reads and writes) data files on disk.

Syntax fyl(Operation, FileNumber [, FileData [,Extra]])

Operation is a number that specifies the action to perform.

FileNumber is an arbitrary number from 1 to 99 that acts as a unique channel number for later Fyl() function calls.

FileData is the name of the file, or the data to write.

Extra is extra information required by certain Operations.

Details The Actions are described below. Unless otherwise mentioned, error conditions (disk full, etc.) are returned numerically in the Sysvar(1) variable. A code of -1 means no error; other codes represent an error (see Appendix C for error code interpretation).

-12 creates directory named FileData (similar to DOS mkdir or md commands)

-11 returns 0 if file opened via Operation 1 contains more data to be read, -1 if at end of the file

-10 returns the contents (up to the first 32K) of the file named FileData previously stored in the book via the Embedded File manager

-9 uncompresses the file named FileData (previously stored in the book via the Embedded File manager) outputting it to the file named Extra; omit Extra to output the file into the directory containing the .ESL currently in use and employ the original file name

-8 uncompresses one or more files from a .ZIP file; specify the full name of the .ZIP in the FileData parameter, the desired output path in Extra, the names of the files to unzip in the 5th parameter, and the switches (if any) in the 6th parameter; for example: `ecode = fyl(-8, 1, "small.zip", "C:\big", "*.*", "-o")`

-7 compresses one or more files into a .ZIP file; specify the full name of the .ZIP in the FileData parameter, the desired files to compress in Extra, and the switches (if any) in the 5th parameter; for example: `ecode = fyl(7, 1, "C:\smaller.zip", "C:\source.txt")`

-6 returns a positive number if a record with the name FileData exists in the linked list file opened via Operation 6 or 7, 0 if the record does not exist

-5 copies the file named FileData to the file named Extra, overwriting Extra if it already exists

-3 returns the date and time of the DOS file named in FileData

-2 returns the length of (number of bytes in) the file named in FileData

-1 returns 0 if file named FileData does not exist on disk, another number if it does

0 closes FileNumber; returns error code, if any

1 opens FileData on channel FileNumber for sequential ASCII text input; returns error code, if any

2 opens FileData on channel FileNumber for sequential ASCII text output; returns error code, if any

3 opens FileData on channel FileNumber for sequential ASCII text append; returns error code, if any

4 opens FileData on channel FileNumber for random binary access with record length Extra (all random files open at the same time must have the same record length); returns error code, if any

6 opens a linked list database named FileData on FileNumber, locked (single user access); if the database does not yet exist, you must include a number from 3 to 504 in the Extra parameter to specify the maximum length (number of characters) in the key (a unique record name by which you will later access data in the database); the key length remains constant for the life of the database; the Extra parameter is ignored if the file already exists; returns error code, if any

7 same as 6, except shared (simultaneous multi-user access)

8 opens ASCII text file FileData on channel FileNumber, returns all text in the file (up to 32K), separating lines with Carriage Return/Line Feed, and closes the file; example: this function is a good substitute for Summit for DOS information boxes...type the following in a Textbox object to load and display the config.sys file at run time `{fyl(8, 1, "C:\config.sys")}`

9 same as 8, except uses Extra as the separator between lines; returns the text in a format compatible for display in Listboxes via ItemList; if Extra is omitted, a single space is used; example: `Listbox(1).ItemList = fyl(9, 1, "C:\ config.sys", chr(13))`

10 same as 8, except reads file as is (useful for non-ASCII files); can also load small (i.e. < 32K) files from Inter/intranet sites (see Examples below); set the Extra parameter to 1 to strip HTML codes; set the Extra parameter to 2 if word wrap is improper (due to lack of CRLF pairs)

11 returns the next line of ASCII text from a sequential file previously opened for input on channel FileNumber via Operation 1; to read from a certain byte number, include the byte number as FileData; returns the text up to the next Carriage Return/Line Feed; sets sysvar(1) to 62 after reaching the end of the file (i.e. when no more data to read)

14 returns record number FileData from a random file previously opened on channel FileNumber via Operation 4

16 returns record with the key name FileData from the linked list previously opened on channel FileNumber via Operation 6

22 writes FileData to a sequential file previously opened on channel FileNumber via Operation 2 or 3 and appends a Carriage Return/Line Feed; optionally specify the byte position via Extra; returns error code, if any

23 same as Operation 22, but does not append a Carriage Return/Line Feed

24 writes FileData to record number Extra in a random file previously opened on channel FileNumber via Operation 4; returns error code, if any

26 writes FileData to the record with the key name Extra in a linked list file previously opened on channel FileNumber via Operation 6; returns error code, if any

28 opens the ASCII text file named FileData, appends Extra to it (along with a Carriage Return/Line Feed), then closes the file; returns error code, if any

31 deletes the file named FileData from the disk; returns error code, if any

36 deletes the record named FileData from the linked list file previously opened on channel FileNumber; returns error code, if any

41 downloads the Inter/intranet file whose URL is specified in FileData and waits for completion; places it into the cache (the cache location is specified via Sysvar(179)); returns the name of the file (including the cache location)

42 same as 41, except does not wait for completion; the only way to know when the download is complete is to compare the length of the file in the cache to its actual, known size

43 uploads the local file named FileData to the site specified in the various FTP entries in the EVEREST.INI (or Sysvar(183) to Sysvar(186); if you want the file to be posted to the site under a different name, include the desired name in the Extra parameter; returns error code, if any

Examples The following A-pex3 programming example quickly loads the contents of the C:\AUTOEXEC.BAT file into the Textbox with IDNumber 1:

```
Textbox(1).Text = fyl(10, 1, "C:\autoexec.bat")
```

The following example downloads the ABC.HTML file from an Internet site:

```
Textbox(1).Text = fyl(10,1,"http://www.xyz.com/abc.html")
```

The following example does the same, and strips the HTML codes:

```
Textbox(1).Text = fyl(10,1,"http://www.xyz.com/abc.html",1)
```

The following example appends the contents of the variable named response to the ASCII text file C:\ANSWERS.TXT (the file is created automatically if it does not yet exist), and reports any errors:

```
ok = fyl(28, 1, "C:\answers.txt", response)
IF ok # -1 THEN ok = mbx("Error " + ok + " " + msg(ok))
```

The following example writes the items in the Listbox with IDNumber 1 into a file on the user records drive (sysvar(56)) named with the current user's unique sequential log on number (sysvar(130)):

```
filename = pth(3, sysvar(56)) + sysvar(130) + ".txt"
filenum = 1                          $$ arbitrary number
ecode = fyl(2, filenum, filename)    $$ open for output
IF ecode # -1 THEN GOTO error
max = Listbox(1).ItemCount           $$ # items in listbox
cnt = 0                              $$ init counter
DO IF cnt < max
  .LookAt = cnt
  ecode = fyl(22, filenum, .Item)    $$ write the item
  cnt++
LOOP
LABEL error
dummyvar = fyl(0, filenum)           $$ close file
IF ecode # -1 THEN
  dummyvar = mbx("Error " + ecode)
ENDIF
```

The following example reads the ASCII text file created via the previous example, and puts its contents in the Listbox with IDNumber 1:

```
filename = pth(3, sysvar(56)) + sysvar(130) + ".txt"
filenum = 1                          $$ arbitrary number
ecode = fyl(1, filenum, filename)    $$ open for input
IF ecode = -1 THEN
  Listbox(1).ItemList = ""           $$ clear previous
  DO
    .AddItem = fyl(11, filenum)      $$ read a line
    IF sysvar(1) # -1 then ecode = sysvar(1): OUTLOOP
  LOOP IF fyl(-11, filenum) = 0      $$ if more data
ENDIF
dummyvar = fyl(0, filenum)           $$ close file
IF ecode # -1 THEN
  dummyvar = mbx("Error " + ecode)
ENDIF
```

The following example searches for all C:\CONFIG.* files, reads them into memory, and writes each into a linked list database named CONFIGS.LL:

```
filename = "C:\configs.ll"
filenum = 1             $$ arbitrary number
namelen = 12      $$ record name length
ecode = fyl(6, filenum, filename, namelen)
IF ecode # -1 THEN GOTO error
cnfgname = ext(41, "C:\config.*")
DO IF len(cnfgname) > 0 $$ do if found a file
  contents = fyl(8, 2, "C:\" + cnfgname)
  IF sysvar(1) # -1 THEN ecode=sysvar(1): OUTLOOP
  ecode = fyl(26, filenum, contents, cnfgname)
  IF ecode # -1 THEN OUTLOOP
  cnfgname = ext(42)    $$ get next file name
LOOP
LABEL error
contents = ""           $$ release memory
dummyvar = fyl(0, filenum)
IF ecode # -1 THEN
  dummyvar = mbx("Error " + ecode, 0)
ENDIF
```

Also see Ext() Function, Pth() Function

Gauge Object

Description Displays an analog representation of a numeric value, either as a needle or bar.

Attributes
Alignment
BorderColor
BorderStyle
Bottom
Caption
CaptionColor
ClickEvent
Comment
Condition
Create
Destroy
DragMode
EdgeSize
EdgeSizeInner
EdgeStyle
Enabled
FillBarColor
FillColor
FillValue
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
GaugeStyle
Height
IDNumber
Initially
InnerBottom
InnerLeft
InnerRight
InnerTop
Left
LightColor
Max
Min
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
Name
Pic
Right
SaveAsObject
SetFocus
ShadowColor

TabOrder
TabStop
Top
Update
Visible
Wallpaper
Width
Zev
ZOrder

Details — You can change the Gauge value displayed by changing the FillValue attribute via A-pex3 programming.

Example — The following A-pex3 code example sets the Gauge with IDNumber 1 to half full (or half empty, whichever outlook you prefer!):

```
range = Gauge(1).Max - Gauge(1).Min
Gauge(1).FillValue = range / 2
```

Also see — Program Object

GaugeStyle Attribute

Applies to	Gauge object	
Description	Sets the appearance of the gauge.	
Settings	0	horizontal bar
	1	vertical bar
	2	180 degree needle
	3	360 degree needle
Also see	FillBarColor	

Gdc() Function

Applies to A-pex3 programming

Description Returns useful information from Windows about the display device.

Syntax gdc(Numeric)

Details This function calls the Windows GetDeviceCaps API function. Use one of the following Numerics to obtain the indicated information:

0	device driver version
2	technology: returns 1
4	width of physical display, in millimeters
6	height of physical display, in millimeters
8	width of display, in pixels
10	height of display, in pixels
12	number of adjacent color bits for each pixel
14	number of color planes
16	number of device-specific brushes
18	number of device-specific pens
20	number of device-specific markers
22	number of device-specific fonts
24	number of entries in the device's color table
26	size of the pdevice internal structure, in bytes
36	clipping capabilities the device supports
38	raster capabilities the device supports
40	relative width of a device pixel used for line drawing
42	relative height of a device pixel used for line drawing
44	diagonal width of a device pixel used for line drawing
88	number of pixels per logical inch along the display width

90	number of pixels per logical inch along the display height
104	number of entries in the system palette
106	number of reserved entries in the system palette
108	color resolution of the device, in bits per pixel

Example The following example warns the user if they are running below 1024 x 768 resolution with 256 colors:

```
IF gdc(8) < 1024 @ gdc(10) < 768 @ gdc(12) < 8 THEN
  txt = "We're sorry.  You need to be running Windows"
  txt = txt + "at 1024 x 768 x 256 resolution, minimum,"
  txt = txt + "in order to view this presentation."
  dummyvar = mbx(txt, 16)
  BRANCH @exit
ENDIF
```

Also see Gsm() Function

GFILL Command

Applies to A-pex3 Xgraphics programming

Description Fills the background of a window (or Picture object) with a graduated color.

Syntax GFILL (Red1, Green1, Blue1, Red2, Green2, Blue2)

Details The GFILL command excels at creating a shadowed background for a window. Via Red1, Green1 and Blue1, you specify the intensity of the three primary colors at the top of the window. Red2, Green2 and Blue2 let you specify the intensity of the colors at the bottom of the window. Each color parameter can range from 0 (minimum color) to 255 (maximum color).

Example The following example fills the window with a graduated blue color:

```
GFILL (0, 0, 0, 0, 0, 255)
```

The following example fills the Picture object with IDNumber 1 with a graduated blue color:

```
was108 = sysvar(108)    $$ save
sysvar(108) = 1         $$ draw on Picture object #1
GFILL (0, 0, 255, 0, 0, 0)
sysvar(108) = was108    $$ restore previous value
```

Notes For proper operation, include a space between GFILL and (.

Due to a Windows limitation, do not use GFILL in a window that contains a BgndPicture.

Also see PAINT

GOSUB Command

Applies to A-pex3 programming

Description Invokes a Program object as a subroutine; execution returns when the subroutine has finished. Specify the Name of the desired Program.

Syntax GOSUB <ProgramObjectName>

Details GOSUB is handy when you want to execute the commands of a Program object. Many authors use GOSUB from the ClickEvent of objects. GOSUB can also be used in a Program to invoke a different Program.

The Program object referenced by the GOSUB command need not be present in the current page; however, Everest can find and execute Programs in the current page more quickly. If the Program object is not in the current page, its SaveAsObject attribute must be enabled (i.e. set to Yes).

Some authors create a page in their book that does nothing except serve as a holder for commonly used Program objects. When needed, they execute the individual Program objects from other pages via the GOSUB command.

Example
```
GOSUB mysubs_program_A
```

Notes To include other A-pex3 commands after the GOSUB command on the same line, separate with a colon and a space. For example:

```
GOSUB routine: tries = tries + 1
```

The nesting depth of GOSUB commands is limited only by available memory. We recommend that you do not nest GOSUBs more than 8 levels deep. Do not used nested GOSUBs if you will be making your project granular.

To GOSUB to a Program object in a different book, prefix the Program object's name with the location (i.e. disk path), book and a semicolon. Program objects in different books should not execute JUMP, BRANCH, CALL or OPEN commands. This feature is not available in granular projects.

During a GOSUB, do not CALL a page that contains a Wait object.

Do not use the RETURN command to return from a GOSUB; instead, use `GOTO @exitprog`.

Also see CALL, GOTO, JUMP, SaveAsObject

GotFocusEvent Attribute

Applies to	Button, Check, Combo, HScroll, Input, Listbox, Mask, Media, OLE, Option, VScroll objects
Description	This event fires when the object receives the focus (the highlight).
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	The focus moves to an enabled object when the user clicks on it, or presses the Tab key to advance to it. Authors often use GotFocusEvent to initialize or update the contents of the object, or change its colors to draw attention to it. When you enter a number or string constant for GotFocusEvent, you are merely telling Everest what event to generate when the object gets the focus. To make use of that event (i.e. detect it and do something useful), you must include a Wait object in your page.
Also see	ClickEvent, LostFocusEvent, SetFocus

GOTO Command

Applies to A-pex3 programming

Description Redirects execution of a Program object to the location of the corresponding LABEL command in the current Program object.

Syntax GOTO <LabelName>

Details The GOTO command tells Everest to look for a LABEL command with a matching <LabelName> in the current Program object, and continue program execution there.

Alternatively, to end execution of the Program object, use the key word @exitprog as the LabelName.

Excessive use of GOTO commands can make your program difficult to understand. When using GOTO, be sure to include a comment to describe its purpose. A comment can be included on any line by prefixing it with $$.

Examples

```
IF name = "user" THEN GOTO bottom  $$ goes to "LABEL bottom"

IF name = "user" THEN GOTO @exitprog$$ keyword
```

Note Compare the GOTO command (which goes to a LABEL in the current program object) with the JUMP command (which goes to a JLabel object in the current page).

Also see GOSUB, JUMP, LABEL

Group Attribute

Applies to Button, Option objects

Description Arranges objects into a set.

Settings 0 to 32,000

Details The Group attribute identifies the objects that make up a set (any arbitrary collection) so they operate as a unit.

OPTION OBJECTS

Typically, Option objects are employed to allow users to choose one of a short list. When the user chooses one, it is selected, and the other members in the same set are automatically deselected.

If all the Options objects in a window belong to one set, simply pick a number between 0 and 32000, and enter it for the Group attribute of all members of the set.

If you have multiple sets of Option objects in one window, assign different Group attribute numbers to each set.

BUTTON OBJECTS

The Group attribute is handy for Button objects when you want to create a button bar. A button bar operates like an old car radio where pressing one button released another.

To make a button bar, put multiple Button objects in a window, set their Group attributes to the same number, and set the HoldDown attribute to Yes. To determine which Button is down, use the GroupChoice or Value attributes.

For non-button bar buttons, set HoldDown to No. If you have both bar and non-bar buttons in the same window, use different Group numbers for each.

Notes Button and Option objects operate independently of each other. Consequently, you can have Button and Option objects with the same Group number.

Also see GroupChoice, HoldDown, Value

GroupChoice Attribute

Applies to Button, Option objects

Description Returns the IDNumber of the object within the Group that is selected. Read-only. Available at run time only.

Details Typically, when a window has several Option or Button objects, you need to know which the user has selected. You could determine this by checking the Value attribute for each object in the Group, but GroupChoice does this for you automatically.

If no object within the group is selected, GroupChoice returns 0.

Example The following example removes the selected Option object from the window:

```
choice = Option(1).GroupChoice
IF choice > 0 THEN ok = Option(choice).Destroy
```

Also see Group, HoldDown

Grouped Attribute

Applies to	Judge object
Description	Determines whether Everest treats objects arranged in a particular Group as a single unit for answer judging, scoring and CMI purposes.
Settings	Yes answer judge Groups as a unit No answer judge each individual object
Details	Most authors enable the Grouped attribute when they have a multiple choice type question page. Typically, such pages contain several Button and/or Option objects that are intended to count as a single question. Enable Grouped to tell Everest to count them as a unit.
Also see	CMIData, Group

Gsm() Function

Applies to A-pex3 programming

Description Returns various useful information from Windows.

Syntax gsm(Numeric)

Details This function calls the Windows GetSystemMetrics API function. Use one of the following Numerics to obtain the indicated information (all measurements are expressed in pixels):

0	width of monitor
1	height of monitor
2	width of arrow bitmap on a vertical scroll bar
3	height of arrow bitmap on a horizontal scroll bar
4	height of window titles
5	width of window frame that cannot be sized
6	height of window frame that cannot be sized
7	width of frame when window has the dialog frame style
8	height of frame when window has the dialog frame style
9	height of scroll box on vertical scroll bar
10	width of scroll box on horizontal scroll bar
11	width of an icon
12	height of an icon
13	width of cursor
14	height of cursor
15	height of single-line menu bar
16	width of window client area for a full-screen window
17	height of window client area for a full-screen window
18	height of Kanji window
19	non-zero if the mouse hardware is installed

20	height of arrow bitmap on a vertical scroll bar
21	width of arrow bitmap on a horizontal scroll bar
22	non-zero if the Windows version is a debugging version
23	non-zero if the left and right mouse buttons are swapped
28	minimum width of window
29	minimum height of window
30	width of bitmaps contained in the title bar
31	height of bitmaps contained in the title bar
32	width of window frame that can be sized
33	height of window frame that can be sized
34	minimum tracking width of window
35	minimum tracking height of window
36	width of rectangle around the location of the first click in a double-click sequence
37	height of rectangle around the location of the first click in a double-click sequence
38	width of rectangles the system uses to position tiled icons
39	height of rectangles the system uses to position tiled icons
40	alignment of pop-up menus
41	handle of the Pen Windows DLL if Pen Windows is installed
42	non-zero if current version of Windows uses double-byte characters

Also see Gdc() Function

hDC Attribute

Applies to	Layout, Picture objects
Description	Contains the Windows handle to Device Context. Read-only. Available only at run time.
Details	The hDC attribute contains the unique number that Windows uses internally to identify the device context of an object. Certain special functions sometimes require this number.
Example	Refer to the example for the Bbt() Function.

HeadingSize Attribute

Applies to	Flextext object
Description	Controls the size of the heading font.
Details	To employ the font size set with HeadingSize, place the desired text between the special codes \H and \h. See the Text attribute for more details.
Also see	FontSize

Height Attribute

Applies to	Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects
Description	Controls the display height of the object.
Details	Specify in units of pixels. The easiest way to adjust the Height attribute of an object at design time is via the VisualPage editor. First, click on the object to focus on it, then point to one of the sizing handles, press and hold down the left mouse button, and drag the mouse. For windows, the Height is controlled via the Layout object. The Height includes the window border, title bar and menu elements.
Also see	Left, Move, Top, Width

HelpAction Attribute

Applies to	Wait object
Description	Specifies the action to perform when the HelpActivator event is triggered.
Double click	Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.
Details	When a Wait object sees that an event code matches the HelpActivator event, it traps that event code, and performs the HelpAction. Most authors employ HelpActivator and HelpAction to display context sensitive help information.
Also see	HelpActivator, HyperAction

HelpActivator Attribute

Applies to	Wait object
Description	Specifies the numeric event code that triggers the HelpAction.
Settings	-32000 to 32000, or a string surrounded by quotes
Double click	Opens event code dialog box. Press the desired key to automatically generate the corresponding event code.
Details	Everest watches the events that occur in your project, and checks if one matches the event code you specify as the HelpActivator. If a match is found, the event is removed from the queue, and Everest performs the HelpAction.
Example	To make an F1 keypress the event that invokes the HelpAction, set the HelpActivator to the event code for F1: 112.
Also see	HelpAction, Wait Object

Hex() Function

Applies to	A-pex3 programming
Description	Returns a number in hexadecimal form.
Syntax	hex(Numeric) Numeric is an integer.
Example	The following calculation stores "10" in the variable named hexval: `hexval = hex(16)`
Also see	Val() Function

Hlp() Function

Applies to A-pex3 programming

Description Invokes the Microsoft Windows help system to provide hypertext help. Returns 0 upon error, any other value indicates success.

Syntax hlp(HelpFile, Action, TopicKey)

HelpFile is the name of the disk file that contains the help. Usually help files have .HLP file name extensions.

Action is a numeric value that designates the operation to perform.

TopicKey is a number or a string (depending on Action).

Details To create your own .HLP file, you will need the Microsoft Help Compiler or equivalent. A discussion of the steps needed to create an .HLP file is beyond the scope of this manual.

Everest's Hlp() function gives you access to the contents of an .HLP file. Set the Action parameter to one of the following values:

1	display help for the topic number specified in TopicKey in the file HelpFile
2	stop using HelpFile; use this to tell Windows you are finished using HelpFile, and let Windows release associated resources
3	display the main help index topic as defined in the [OPTIONS] section of the .HPJ help project file
4	display Windows' Help-on-Help (WINHELP.HLP)
5	set the index for HelpFile to TopicKey
257	display help for the topic named in TopicKey in the file HelpFile, or "invalid keyword" if not found
261	display help search window

Example The following one-line program opens the help search window for the notepad application:

```
dummyvar = hlp("C:\windows\notepad.hlp", 261)
```

Also see Flextext Object, Hyperhlp Object

HoldDown Attribute

Applies to	Button object
Description	Determines whether a button remains down after being depressed.
Settings	Yes keep button down after press No button automatically springs up after press
Details	Set HoldDown to Yes when you want the button setting to "stick" when the user clicks on it. The user can release the button by clicking on it again. Coupled with the Group attribute, HoldDown can be used to create button bars.
Also see	Group, Value

HScroll Object

Description The HScroll object is a horizontal slider bar with pointer that the user can adjust.

Attributes
Answers1
Answers2
AntIncorrect1
Bottom
ChangeEvent
CMIData
Comment
Condition
Create
Destroy
DragMode
Enabled
GotFocusEvent
Height
IDNumber
Ignore
Initially
JudgeVar
Judgment
LargeStep
Left
LostFocusEvent
Max
Min
MousePointer
Move
Name
Preset
ResponseVar
Right
SaveAsObject
SetFocus
Step
TabOrder
TabStop
Top
Tries
Update
Value
Visible
Width
Zev
ZOrder

Details Many authors employ an HScroll object to allow the user to choose from a range of numeric values. You can assign the value represented by the left and right edge of the bar via Min and Max respectively.

Also see VScroll Object

HValue Attribute

Applies to	Layout object
Description	Determines the position of the pointer on the window's horizontal scroll bar.
Settings	0 to VirtualWidth
Also see	Scrollable, VirtualWidth, VValue

hWnd Attribute

Applies to	Animate, Button, Check, Combo, Frame, Gauge, HScroll, Input, Layout, Listbox, Media, Option, Picture, SPicture, VScroll objects
Description	Contains the Windows handle to an object. Read-only. Available only at run time.
Details	The hWnd attribute contains the unique number that Windows uses internally to identify an object. Certain special functions sometimes require this number.
Example	Refer to the example for the Mci() Function.

HyperAction Attribute

Applies to	Hyperhlp object	
Description	Specifies the type of operation to perform on the HyperFile help file.	
Settings	1	display help for the topic context number specified in HyperTopic
	2	tells the Windows help system that help is no longer needed (releases memory)
	3	displays the main help index topic as defined in the [OPTION] section of the help project file (.HPJ)
	4	display "help on help"; loads the Windows WINHELP.HLP file and displays the "Using Help" index topic
	5	makes HyperTopic the index topic
	257	displays help for the topic named in HyperTopic
Also see	HyperFile, HyperTopic	

HyperFile Attribute

Applies to	Hyperhlp object
Description	Specifies the name of the disk file that contains the help information.
Double click	Opens file dialog box. Double click on the file you want.
Details	Create the help file (which usually has a file name extension of .HLP) with the Microsoft Windows Help Compiler.
Also see	Hlp() Function, HyperAction

HyperHlp Object

Description Links to the Microsoft Windows help system.

Attributes Condition
HyperAction
HyperFile
HyperTopic

Details Use the HyperHlp object to provide a user with hypertext style access to information. Via the HyperHlp object, you can invoke the help system that is included with Windows.

The Windows help system employs .HLP files. You can create .HLP files with the Microsoft Windows Help Compiler (not included with Everest).

To build hypertext without the Windows help system, refer to the Flextext Object.

HyperTopic Attribute

Applies to	HyperHlp object
Description	Specifies the number or name of the help topic to display.
Also see	HyperAction

Ibx() Function

Applies to A-pex3 programming

Description Displays a message window with one fill-in style field for user input. Returns the input made by the user.

Syntax ibx(Message, Caption [, Default])

Message is a character string to display inside the window; typically it is employed to prompt the user for input.

Caption is a character string to display in the title bar of the window.

Default is a character string to initially display in the fill-in field.

Details OK and Cancel buttons are included in the window. If the user chooses Cancel, an empty string is returned.

Example The following A-pex3 example asks the user to enter a word for which to search. The example then attempts to find that word in the Textbox with IDNumber 1, and, if found, scrolls the text to that position:

```
search4 = ibx("Enter a search word", "Search", search4)
IF len(search4) > 0 THEN
  foundat = lwr(Textbox(1).Text) * lwr(search4)
  IF foundat > 0 THEN Textbox(1).SelStart = foundat
ENDIF
```

Also see Mbx() Function

Icon Attribute

Applies to	Layout object
Description	Loads an icon for display when the window is minimized. Write-only. Not saved between user sessions.
Details	Set the Icon attribute to the name of the .ICO file that contains the desired icon. When the window is minimized, this icon will be displayed in place of the default one supplied by Everest.
Example	The following A-pex3 example loads an icon for the current window: window(0).icon = "everest.ico"
Also see	WindowState

IDNumber Attribute

Applies to — Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Line, Listbox, Mask, Media, Menu, OLE, Option, Picture, Shape, SPicture, Textbox, Timer, VScroll objects

Description — Specifies a number that, for a given window, uniquely identifies the object within its class (Textbox, Picture, etc.). Accessible at design time only.

Settings — 1 to 99

Details — An IDNumber is assigned automatically by Everest when you create the object. In most cases, on a given page, each object in a particular class (Textbox, Picture, etc.) has a unique IDNumber.

When your page is executed, an object that has a unique IDNumber will be ADDED to the window. For special purposes, you can change the IDNumber to match that of another object in the same class. An object that has the same IDNumber as a previous object will REPLACE that object in the window.

Two objects that have the same IDNumber appear as separate objects while editing. They count as two objects towards the limits of number of objects per class and page.

Examples — When you refer to object attributes in A-pex3 programming, include the IDNumber (not a number in the Name, if any) inside the parentheses; for example:

```
Textbox(5).Top = 100
```

sets the top of the Textbox with IDNumber 5 to window location 100.

To reference a Layout object (which actually has no IDNumber attribute) in A-pex3, use an IDNumber of 1.

To set an attribute of a group of objects with consecutive IDNumbers, use the following syntax (the word "to" must be in lower-case):

```
Shape(1 to 5).Visible = 0
```

Also see — Condition, Erase Object, Name, Obj() Function

IF Command

Applies to A-pex3 programming

Description Tests a condition and performs one or more actions.

Syntax **SINGLE LINE STYLE**

IF <Condition> THEN <Action> [ELSE <Action>]

BLOCK STYLE

```
IF <Condition> THEN
  <Action>
ELSEIF <Condition> THEN
  <Action>
ELSE
  <Action>
ENDIF
```

DO...LOOP

Refer to the DO and LOOP commands.

Details Use IF to perform an action, such as a calculation or command, under certain conditions.

<Condition> has the form <Operand1> Relational Operator> <Operand2>. For example:

```
IF response = "baseball" THEN strike = 1

IF Input(1).Text =E= "X" THEN flag = 1 ELSE flag = 0
```

MULTIPLE CONDITIONS

You can test multiple conditions by separating them with either @ (which represents OR) or & (which represents AND). Everest evaluates multiple conditions from left to right. Conditions cannot be grouped (such as with parentheses). An example that tests if two conditions are true:

```
IF x1 < 50 & y1 < 25 THEN topleft = 1
```

BLOCK STYLE

Block style IF commands are handy when you want to check several conditions, and perform a different action for each. The optional ELSE clause performs an action when no previous condition was true. An example:

```
IF keypress = 112 THEN
  helpflag = 1
  BRANCH help
ELSEIF keypress = 113 THEN
```

```
  BRANCH menu
ELSEIF keypress = 114 THEN
  IF count > 0 THEN count = 0
  BRANCH next
ELSE
  JUMP top
ENDIF
```

The commands you want to perform inside the block style IF must be indented with a least one space.

Also see Operators

Ignore Attribute

Applies to — Button, Check, Combo, HScroll, Input, Listbox, Mask, Option, VScroll objects

Description — Specifies one or more possible responses that Everest should ignore during answer judging via the Judge object.

Details — Rather than judge a user's response as either correct or incorrect, it can sometimes be helpful to simply ignore the response. Ignored responses do not count for scoring purposes, are not recorded by the CMIData feature, and do not decrement the Tries counter.

To specify Ignore answer matches, use the same syntax you would for the Answers1 attribute.

When Everest finds a match for the user's response in the Ignore answer list, it sets the value of Judgment and the JudgeVar (if any) to a null string. The ignored response is returned in the ResponseVar (if any). You can use this information to display feedback, if desired.

To let the user try again, jump back to the Wait object, use another Wait object in the page, or let Everest run the page to the end, in which case it searches for the Wait object nearest the end of the IconScript and automatically jumps to it.

An easy way to determine if active question objects remain from the most recent Judge is to examine Sysvar(4). When a Judge object is processed, Everest counts the number of question objects that are skipped (because of a match with the Ignore answer list), or which were answered incorrectly and have additional Tries remaining, and stores the total in Sysvar(4). Fields that have no limit on number of Tries are not counted.

Example — If the range of possible responses on a multiple choice question is A to D, if the user enters the letter E, you might wish to simply ignore it and allow another attempt. You would enter:

```
<a|>d
```

as the Ignore attribute to ignore any responses outside the range A to D.

Also see — Answers1

Include Object

Description	With the Include object, you can combine other pages with the current one when the page is run. Basically, the Include object calls another page as a subroutine; that means it processes the included page as if its objects were part of the page that has the Include.
Attributes	Comment Condition IncludePage Name
Also see	CALL, GOSUB

IncludePage Attribute

Applies to	Include object
Description	Specifies the name of the page whose objects you want to combine into the current page at run time. Similar in concept to a subroutine call.
Double click	Opens page name dialog box. Double click on the name of the page to include.
Details	The IncludePage must not contain a Wait object. To help avoid IDNumber conflicts, consider assigning unusual IDNumbers to the objects on a page to be used as an IncludePage. For example, you might reserve IDNumbers 90 to 99 in your project for just such usage.
Example	If many of the pages in your project will have a row of navigation buttons, you can create the buttons on one page, and refer to that page from others via the IncludePage feature. Later, if you need to change the buttons, you only need to make the change to that one page.
Notes	If the IncludePage is located in a different book, prefix the page name with the disk path, book and a semicolon. IncludePages in different books should not contain JUMP, BRANCH or CALL commands.
Also see	EraseFromID, IDNumber

Indent Attribute

Applies to	Input, Textbox objects
Description	Controls the size of the empty area between the four edges of the object and the text.
Settings	1 to 255 (pixels)
Also see	BorderStyle

Ini() Function

Applies to A-pex3 programming

Description Reads and writes Windows .INI format files.

Syntax ini(Operation, FileName, Section, Topic, String [, Length])

Operation is 1 to read from a file, or 2 to write to it.

FileName is the name of the .INI file to use.

Section is the name of the section in the .INI file (do not include the [ ]).

Topic is the name of the element within the section (do not include the =).

When Operation is 1, String is the default string to return if the Topic is not found.
When Operation is 2, String is the character string to write into the .INI file.

When Operation is 1, Length is the maximum number of characters to read from the .INI file. When Operation is 2, Length can be omitted.

Details For the curious: this function calls the Windows API GetPrivateProfileString and WritePrivateProfileString functions.

Example The following example determines if Microsoft Video for Windows has been installed on the computer:

```
file = "win.ini"
sec = "mci extensions"
topic = "avi"
string = "not found!"
      length = 8
      vfw = ini(1, file, sec, topic, string, length)
      IF vfw =E= "avivideo" THEN
        $$ found it
      ELSE
        $$ not found
      ENDIF
```

Also see Fyl() Function

Initially Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Line, Listbox, Mask, OLE, Option, Picture, Shape, SPicture, Textbox, Timer, VScroll objects

Description — Tells Everest how to set certain attributes at run time if an object (of this class and IDNumber) does not already exist in the window. Read-only at run time.

Settings

0	Visible = No, Enabled = No
1	Visible = Yes, Enabled = No
2	Visible = No, Enabled = Yes
3	Visible = Yes, Enabled = Yes
7	Visible = Yes, Enabled = Yes, and set focus to this object
9	Visible = Yes, Enabled = No, and force visual update of this object
11	Visible = Yes, Enabled = Yes, and force visual update of this object
15	Visible = Yes, Enabled = Yes, set focus, and force visual update

Details — The Initially attribute lets you control how the object appears when it is added to a window. In most situations, you should use the default setting of 3. The Visible = No and/or Enabled = No settings are useful only if you later change the value of the Visible and/or Enabled attributes at run time via A-pex3 programming.

If you want the cursor (the focus) to begin at a particular object when the user first sees the page, set Initially to 7 or 15 for that object. A setting of 7 or 15 is available only for those objects that allow SetFocus.

Due to the way Microsoft Windows visually refreshes a window, the order in which the objects appear at run time may not match their order in the Book Editor. For example, a Flextext object might plot after other objects have, even though the Flextext object comes first in the Book Editor. You can force Windows to visually update a particular object upon encountering it in the Book Editor. To do so, set Initially to either 9, 11 or 15.

For objects that do not have an Enabled attribute (such as Shapes and Lines), Everest ignores the Enabled portion of the Initially attribute.

For objects that do not support an Update attribute, Everest ignores the forced update portion of the Initially attribute.

Example — In the following example, the location of the Textbox with IDNumber 1 depends on the value in variables xoffset and yoffset. If Initially is set to 3, the user will see the object appear at its design time location, And then jump to the xoffset, yoffset location. To avoid this distracting movement, set Initially to 0, and use code similar to the following in a Program:

```
Textbox(1).Move = reg(xoffset, yoffset)
Textbox(1).Visible = 1
Textbox(1).Enabled = 1
Textbox(1).ZOrder = 0   $$ optional
```

Also see — Enabled, SetFocus, Update, Visible, Zev, ZOrder

InnerBottom Attribute

Applies to	Gauge object
Description	Sets the size of the inner area at the bottom of the Gauge, relative to its edge.
Details	Specify in units of pixels.
Also see	GaugeStyle, InnerTop

InnerLeft Attribute

Applies to	Gauge object
Description	Sets the size of the inner area at the left of the Gauge, relative to its edge.
Details	Specify in units of pixels.
Also see	GaugeStyle, InnerRight

InnerRight Attribute

Applies to	Gauge object
Description	Sets the size of the inner area at the right of the Gauge, relative to its edge.
Details	Specify in units of pixels.
Also see	GaugeStyle, InnerLeft

InnerTop Attribute

Applies to	Gauge object
Description	Sets the size of the inner area at the top of the Gauge, relative to its edge.
Details	Specify in units of pixels.
Also see	GaugeStyle, InnerBottom

Input Object

Description The Input object is a "fill-in-the-blank" style user interaction field.

Attributes
AdjustResponse
Alignment
Answers1...Answers8
AntIncorrect1
AntIncorrect2
BackColor
BorderStyle
Bottom
ChangeEvent
CMIData
Comment
Condition
Create
Destroy
DragMode
EOFEvent
Enabled
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
ForeColor
GotFocusEvent
Height
IDNumber
Ignore
Indent
Initially
InputTemplate
JudgeVar
Judgment
Left
LostFocusEvent
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
MultiLine
Name
PassChar
Position
Preset
ResponseVar
Right
SaveAsObject

SelLength
SelStart
SelText
SetFocus
TabOrder
TabStop
Text
TextLength
Top
Tries
Update
Visible
Width
WordWrap
Zev
ZOrder

Details

Employ an Input object to obtain a typed response from the user. The response is returned in the variable whose name you enter for the ResponseVar attribute.

To judge the user's response for accuracy, place answer specifications in the Answers attributes. The answer judgment is returned in the variable whose name you enter for the JudgeVar attribute.

There is a known bug in the Input object that can cause the cursor to appear in the wrong place when a vector font is used.

Also see

Answers, Combo Object, JudgeVar, Mask Object, ResponseVar

InputTemplate Attribute

Applies to Input, Mask objects

Description Determines the characters that the user is allowed to type at the corresponding position in the field.

Details **INPUT OBJECT**

For an Input object, the InputTemplate string consists of the following symbols (shown with the input they allow):

Space	no input/change at this position
?	any character
#	0-9, minus, space and decimal
9	0-9
@	0-9, space
A	A-Z (upper case)
a	a-z (lower case)
B	A-Z, space
b	a-z, space

If there is room in the_Input object for more characters than specified in the InputTemplate, the last symbol of the InputTemplate applies to the additional characters.

MASK OBJECT

For a Mask object, the InputTemplate string consists of the following symbols:

#	allows 0 to 9
.	decimal point
,	comma (thousands separator)
:	colon (time separator)
/	slash (date separator)
&	any character (ANSI 32 to 126 and 128 to 255)
A	a to z, A to Z, or 0 to 9
?	a to z or A to Z
\	treat next character literally
Other	literal

For a Mask object, in the field itself, the PromptChar is displayed in place of the InputTemplate character to alert the user that input is expected.

Notes InputTemplate is limited to a maximum of 50 characters for an Input object and 64 characters for a Mask object.

Also see Format, MaxLength, TextLength

Instance Window

The Instance Window is accessible from the Attributes Window's Options pull-down menu. The Instance (or Instance of...) Window helps you find a master object to either instantiate or copy.

WHAT IS OBJECT INSTANCING?

Each item, such as a Textbox or Button, you create on a page, is known as an object. In typical projects, most objects are used just once. However, often a few appear on several pages. Identical user "navigation" buttons, such as Next, Back, etc., often appear on many pages.

Everest's object instancing feature lets you create a master object, and reuse it on many pages. If you make a change to a master object, the change appears everywhere the object is instantiated (used). This approach saves you time when you later change your book.

CREATE A MASTER OBJECT

Before you can either instantiate or copy an object, you must save it as a master. If the drop down list in the Instance window is empty, it is because there are no master objects (of this particular class, such as Button) in the book. To save an object as a master, create a page and the object normally, but enable the object's SaveAsObject attribute. You might also want to rename the object to something more meaningful, such as "back_button." Then save the page.

INSTANTIATING

To use the master object later, first create an object normally on the page, then choose "Instance of..." from the Attributes window pull-down menu. Your master objects of the same class (Button, Textbox, etc.) will appear in the list. Double click on the one you want to instantiate.

You should use some caution changing the attributes of instantiated objects. For more information, refer to the instructions for the Attributes editor.

UN-INSTANTIATING

If, for some special reason, you later want to sever the attachement between an instantiated object and the master, you must do two things: rename the instantiated object, and disable its SaveAsObject attribute.

COPYING

If you do not want to establish a link between the object and the master, but simply want to copy it, in the Instance window, click once on the name of the desired object, then click on the Copy button.

Instructions

Most procedural ("how-to") topics are thoroughly documented in the Everest Tutorial and Design Guide books. Brief instructions on how to use the various editing windows and dialog boxes of the AUTHOR program is supplied in the following help topics:

Internet Simulator Window

The INet Simulator can be accessed from the Author window's Run pull-down menu. The INet Simulator simulates the apperance of running your project over the Internet and intranets. It is a handy way to test the execution speed of your project. You can use the INet Simulator even if you do not have Internet or intranet access. You can choose from a variety of simulated modem speeds.

[Or, if you simply want to try running something real over the actual Internet, here's how. With your computer ready to connect to the Internet, from the main Author windows Run menu, choose Start At, click the Direct button, and enter `http://www.insystem.com/evdemo/@start;@start` which will run the current Everest demo from our Web site.]

Page at which to Start Simulation

The entire project you want to test run must be located within a single subdirectory on your local hard drive. In this column, select the drive, directory, book and page at which to begin running the simulation. During the test run, Everest will use this location as the "file server" for all files your project would have normally referenced on the Internet/intranet.

Double click on a book to load the names of the pages into the combo box below. Note: this feature is not supported for .ZIPped books...youll need to enter the page name (usually @start) manually.

Cache Location

In order to perform an accurate simulation, Everest actually transfers your project's files to the location you specify for the download cache. Enter the disk path of a location that has enough room to hold all the files of your project, and contains no files that cannot be overwritten. The Cache Location must be different than the location of your project. **NOTE: since it is possible that files in the Cache Location will be overwritten, be sure the location you specify contains no files you need to preserve.** A RAM disk makes the ideal cache.

Simulated Modem Kbaud Rate

Choose one of the modem speeds from the list provided, or enter one of your own. This lets you observe how different modem speeds will impact the execution speed of your project. If your project runs too slowly, consider using the Project Packager to compress it into .ZIP form, and/or to make it granular.

Clear Download Cache

Everest leaves downloaded files in the cache until: 1) you exit the program, or 2) you manually clear the cache. Enable this feature if you are re-starting a test run, and do not want Everest to employ files downloaded from a prior test run.

InvalidEvent Attribute

Applies to	Mask object
Description	Event code to generate, or programming to perform, when the user's entry does not match that specified by the InputTemplate.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	When you enter a number or string constant for InvalidEvent, you are merely telling Everest what event to generate when the user clicks on the object. To make use of that event (i.e. detect it and do something useful), you must include a Wait object in your page. To determine the position of the invalid character, refer to the value of sysvar(148) immediately after the InvalidEvent fires.
Example	Some authors use the InvalidEvent to detect when the user has entered an illegal character, and display a warning message.

Item Attribute

Applies to	Combo, Listbox objects
Description	Sets or returns the text stored in the active item. Accessible via A-pex3 programming only.
Details	The active item is set via the LookAt attribute. Always set LookAt to the desired value before using Item.
Example	The following example sets the variable named last equal to the text of the last item in the Listbox with IDNumber 1:

```
Listbox(1).LookAt = Listbox(1).ItemCount - 1
last = Listbox(1).Item
```

Notes	After adding/changing items at run time, we recommend that you do not change appearance attributes of the object (such as FontSize); doing so might reset the item list back to its original state. This is due to a bug in the MicroHelp controls that drive these objects.
Also see	ItemCount, LookAt, Tagged

ItemAlignment Attribute

Applies to	Listbox object	
Description	Controls the horizontal justification of text in the list.	
Settings	0	left justify
	1	right justify
	2	center

ItemColor Attribute

Applies to	Combo, Listbox objects
Description	Controls the foreground color of the items in the list.
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.

ItemCount Attribute

Applies to	Combo, Listbox objects
Description	Returns the number of items in the list. Read-only and available at run time only.
Also see	LookAt, TaggedCount, TopIndex

ItemIndex Attribute

Applies to	Combo, Listbox objects
Description	Determines the active item in the list. Available at run time only.
Details	The items are numbered starting with 0 at the top of the list. For a Listbox with a TagStyle of 0 (single select), use ItemIndex to read and set the highlighted item. To remove the highlight entirely, set ItemIndex to -1.
Also see	ItemCount, LookAt, TopIndex

ItemList Attribute

Applies to Combo, Listbox objects

Description Specifies a group of items to be displayed in the list. Write only.

Details Enter the list of items, starting with a unique character used as a separator between the items. ItemList replaces the current list of items (if any) in the object.

To add individual items at run time, use the AddItem attribute.

To have the items appear in multiple columns (i.e. when ColCount is non-zero), the ColChar character must appear within the ItemList at the places you want Everest to break the text into columns.

Examples To create a list of National League East baseball teams, enter the following for the ItemList Attribute:

```
;Mets;Phillies;Expos;Marlins;Braves
```

Sometimes it can be handy to load a Listbox from an ASCII text file (up to 32Kbytes) on disk. To do so, set ItemList to something like:

```
{fyl(9, 1, "C:\config.sys", chr(13))}
```

To see the results of an embedded expression such as the example above, run a Preview.

To display a list of inventory items in a two-column Listbox, set ColCount to 2, ColChar to 44 (a comma) and ItemList to:

```
;pencils,23;pens,14;pads,5
```

To display, at run time, a list of items stored in a text file in the style commonly referred to as "comma delimited format," set ColCount to the proper number of columns, ColChar to 44 (a comma) and ItemList to something like:

```
{fyl(9, 1, "C:\mylist.txt", chr(13))}
```

Also see AddItem, Item, Pik() Function, RemoveItem, Style, Sorted, TaggedList

Iterations Attribute

Applies to	Animate object	
Description	Determines the number of times the animation is displayed (repeated).	
Settings	0	continuous repeat
	1 to 1000	number of times to repeat
Also see	AnimFile, EndFrame, Play	

JLabel Object

Description — The JLabel object acts as a branching destination for an A-pex3 JUMP command.

Attributes — Comment
Name

Details — Normally, Everest executes the objects of your page from top to bottom. Under certain conditions, you may want to vary the execution order, and jump to a certain location in the page. To do so, drag a JLabel object from the ToolSet to the desired location in the Book Editor.

To jump to the JLabel, use the A-pex3 JUMP command. You can place the JUMP command in a Program.

Upon performing a JUMP to a JLabel, Everest continues executing the page at the object that follows the JLabel.

Example — To jump to a JLabel with a Name attribute of "page1_jlabel_A" use the following A-pex3 command:

```
JUMP page1_jlabel_A
```

Most authors change the Name attribute of JLabel objects to make them more descriptive than the name Everest assigns the objects automatically.

Notes — The JLabel object cannot be toggled off.

Also see — JUMP

Judge Object

Description — Designates the place in the page where Everest checks user responses to previous question objects for accuracy. The Judge object stores a value in each object's Judgment attribute that indicates the result of the accuracy check, and also puts the user's response in the ResponseVar.

Attributes
Comment
DisableObjs
Grouped
Name

Details — A Judge object is needed in your page only if both of the following are true:

1) the page has question objects; these include Button, Check, Combo, HScroll, Input, Listbox, Mask, Option and VScroll objects.

2) you want to compare the user's responses to the Answers attributes specified with each interactive object, or easily retrieve the user's response in the ResponseVar.

LOCATION OF JUDGE OBJECT

Most authors place one or more question objects in the page, followed by a Wait object (which allows the student a chance to respond), and then a Judge object.

ENCOUNTERING THE JUDGE OBJECT

Answer judging does not occur until Everest encounters a Judge object as it executes the page. Use either of two ways to trigger this encounter:

1) enter the desired event code(s) in the Wait object's JudgeActivator attribute; for example, if you want judging to occur when the user presses the F2 key, set JudgeActivator to 113.

2) insert a JLabel in the page before the Judge object, and employ the A-pex3 JUMP command to branch to it.

WHICH ARE JUDGED?

When the Judge object is encountered, Everest judges all active question objects located before that Judge object in the page, and after the previous Judge object (if any).

AFTER JUDGING

After judging, Everest continues executing the page normally. Most authors place feedback (such as Textbox objects) after a Judge object. The Judgment attribute can be used in programming to determine the judgment rendered.

EXTRA TRIES

To let the user try again, jump back to the Wait object, use another Wait object in the

page, or let Everest run the page to the end, in which case it searches for the Wait object nearest the end of the page and automatically jumps to it.

An easy way to determine if active question objects remain from the most recent Judge is to examine Sysvar(4). When a Judge object is processed, Everest counts the number of question objects that are skipped (because of a match with the Ignore answer list), or which were answered incorrectly and have additional Tries remaining, and stores the total in Sysvar(4). Fields that have no limit on number of Tries are not counted.

Also see Answers1, JudgeActivator, Judgment, Tries

JudgeActivator Attribute

Applies to	Wait object
Description	Specifies one or more event codes that tell Everest to search for the next Judge object in the page (and, if found, perform answer judging of user responses).
Settings	-32000 to 32000, or a string surrounded by quotes
Details	The JudgeActivator attributes work slightly differently that the other xxxActivator attributes. Everest watches the events that occur in your project, and checks if one matches the event code you specify. If a match is found, the event is removed from the queue, and Everest's processing of the page jumps to the next Judge object. Answer judging is performed, and execution of the page continues from there. In the typical page, authors place objects that provide feedback or perform branching (Textbox, Program objects, etc.) after the Judge object.
Also see	Judgment

JudgeVar Attribute

Applies to	Button, Check, Combo, HScroll, Input, Listbox, Mask, Option, VScroll objects
Description	Indicates the name of the variable in which to store the answer judging results.
Details	NOTE: The JudgeVar attribute is provided mainly for compatibility with the Summit Authoring System for DOS. In Everest, most authors employ the Judgment attribute instead. Upon answer judging via a Judge Object at run time, Everest stores a numeric value, or a null string, in the variable you enter as the JudgeVar. The number represents the line of anticipated answers that contained a match. Authors often use this information to perform customized feedback and scoring via subsequent A-pex3 programming. See the table found with the Judgment attribute for a description of the possible values Everest will store in the JudgeVar variable.
Notes	Do not surround the variable name with { }. Everest does not require that you specify a JudgeVar for proper operation of answer judging.
Also see	Answers1, Condition, Judge Object, ResponseVar

Judgment Attribute

Applies to — Button, Check, Combo, HScroll, Input, Listbox, Mask, Option, VScroll objects

Description — Returns the answer judgment rendered by the most recent Judge object. Read-only. Available at run time only.

Details — Upon answer judging via a Judge Object at run time, Everest stores a numeric value, or a null string, in the Judgment attribute. The number represents the line of anticipated answers that contained a match. Authors often use this information to perform customized feedback and scoring via subsequent A-pex3 programming.

The following table shows the possible values Everest will store in the Judgment attribute, and the meaning of each:

Value	Meaning
1	user's response matched Answers1
2	user's response matched Answers2
3	user's response matched Answers3
4	user's response matched Answers4
5	user's response matched Answers5
6	user's response matched Answers6
7	user's response matched Answers7
8	user's response matched Answers8
-1	user's response matched AntIncorrect1
-2	user's response matched AntIncorrect2
null string	user's response matched Ignore
0	user's response matched nothing

Example — The following A-pex3 programming example demonstrates how you can display a variety of custom feedback messages within a Textbox with IDNumber 3 for a hypothetical question page containing two Input objects. This programming would appear within a Program object positioned after a Judge object:

```
IF sysvar(175) = 0 THEN       $$ most recent judgment
  Textbox(3).Text = "Neither response is correct!"
ELSEIF sysvar(175) = 1 THEN
  IF Input(1).Judgment > 0 THEN
    Textbox(3).Text = "Question 1: answered correctly."
  ELSEIF Input(2).Judgment > 0 THEN
    Textbox(3).Text = "Question 2: answered correctly."
  ENDIF
ELSE
  Textbox(3).Text = "You answered both correctly!"
ENDIF
```

Also see — Answers1, Condition, Judge Object, JudgeVar

JUMP Command

Applies to A-pex3 programming

Description Alters the normal top-down page run-time execution order.

Syntax JUMP <JLabelName>

Details Use JUMP when you want execution to continue at a different object in the page (i.e. to "jump" to a different location).

JLabelName is either the Name of the JLabel object in this page at which to continue execution, or it is one of the following key words:

@proceed Causes execution to continue with the next object in the page. Authors often use this in an xxxAction attribute to force page execution to continue after performance of another command. For example, you might set NextAction to

```
GOSUB routine: JUMP @proceed
```

@wait Tells Everest to search backward in the page, starting with the current object, until it finds a Wait object, and wait there.

Contrast JUMP with GOTO, which causes execution to continue within the same Program object.

Example

```
JUMP page1_jlabel_A
```

Also see BRANCH, GOTO, LOOP

JumpPointer Attribute

Applies to Flextext object

Description Controls the appearance of the mouse cursor while the cursor is positioned over a jump word. Accessible only at run time via A-pex3 programming.

Settings

0	default
1	arrow
2	cross-hairs
3	I-beam
4	icon
5	N, S, E, W arrows
6	NE, SW arrows
7	N, S arrows
8	NW, SE arrows
9	W, E arrows
10	up arrow
11	hourglass
12	"not allowed" symbol
13	hand
14	arm
15	target
16	wand
17	boy
18	girl
19	pencil
20	lightning
21	key
22	telephone
23	question mark

Also see MousePointer, NormalPointer, PopupPointer

Key() Function

Applies to A-pex3 programming

Description Performs various keyboard related operations.

Syntax key(Operation [, Value])

Operation is a number that specifies the action to perform.

Value depends on Operation.

Details Use one of the following Operations:

-1 check the Windows event queue; returns 0 if no event is pending; non-zero means an event is waiting; to check the Everest event queue (which is different than the Windows event queue) use Operation 2

0 re-enable event handling that was disabled with Operation 1, 2, 3, ext(27) or ext(28)

1 disable normal event handling so that all events are trapped into the Everest event queue (for access via Operations 2 and 3); WARNING: disabled event handling may be confusing to the user, and may make your project difficult to debug; use at your own risk since once you disable event handling the only way to enable it again is via key(0)

2 poll for an event from the Everest event queue; returns next event code or 0 if none in queue; does NOT remove event from queue; disables normal event handling; see warning with Operation 1

3 wait for event in Everest event queue; waits for an event to be put in the queue and returns its event code; disables normal event handling; see warning with Operation 1

4 trigger Activator; manually triggers the Action associated with an Activator that matches Value as specified in a Wait object; returns 0 if no Activator was triggered

5 convert event to ASCII; returns the equivalent numeric ASCII code for an event code specified in Value; if no equivalent numeric ASCII code exists, returns -Value

6 return description; returns a text description of the event code specified in Value; use for keypress event codes only

7 stuff keys; appends keypresses in Value to the Windows event queue (just as if the user had pressed the keys)

8 same as 7, except waits for the application to process the keys before continuing

9 calls the Windows API GetQueueStatus function; pass desired flags in Value parameter

STUFF KEY SYNTAX

For Operations 7 and 8, you specify the keys to stuff via a string of characters in the Value parameter. To stuff letters and numbers, simply use the letters and numbers themselves. Certain other keys must be surrounded by { }. Here is a list:

Backspace	{BKSP}
Break	{BREAK}
Caps Lock	{CAPSLOCK}
Clear	{CLEAR}
Del	{DEL}
Down arrow	{DOWN}
End	{END}
Enter	{ENTER} or ~
Esc	{ESC}
F1 to F12	{F1} to {F12}
Help	{HELP}
Home	{HOME}
Ins	{INSERT}
Left arrow	{LEFT}
Num Lock	{NUMLOCK}
Page down	{PGDN}
Page up	{PGUP}
Print screen	{PRTSC}
Right arrow	{RIGHT}
Scroll Lock	{SCROLLLOCK}
Tab	{TAB}
Up arrow	{UP}

SHIFT KEYS

To press the Shift, Ctrl or Alt keys with a given key, precede the key with one or more of the following codes:

Shift	+
Ctrl	^
Alt	%

MULTIPLE KEYS WITH SHIFT

To specify the one or more of Shift, Ctrl or Alt keys should be held down while several other keys are pressed, enclose the key in ( ). For example, to hold down Shift while stuffing EVEREST use key(7, "+(EVEREST)").

REPEATED KEYS

To repeat a key, include a space and A number after the key, and surround with { }. For

example, to press s 5 times, use key(7, "{s 5}").

Example

The following example disables normal event handling, waits for an event, then re-enables normal event handling:

```
dummyvar = key(1)       $$ disable handling
event = key(3)          $$ wait
dummyvar = key(0)       $$ enable handling
```

Also see

Ext(5) Function, Shl() Function, Stf() Function

LABEL Command

Applies to	A-pex3 programming
Description	Marks a location in a program object that is the destination of a GOTO command.
Syntax	Label <LabelName>
Details	Place the LABEL at a location in the program object to which you want processing to be redirected via the GOTO command. Do not indent a LABEL command; it must appear at the left edge of the program code editor screen. This also means you may not place a LABEL inside an IF block or DO...LOOP construct.
Notes	Everest ignores anything after the LABEL on the same line.
Also see	GOTO, Program Object

LargeStep Attribute

Applies to	HScroll, VScroll objects
Description	Sets the amount by which to change a scroll object's value when the user clicks in the area between the scroll box and scroll arrow.
Also see	Step

LastAdded Attribute

Applies to Combo, Listbox objects

Description Returns the index number of the last item added via AddItem. Read-only. Available at run time only.

Details LastAdded is especially handy when Sorted is set to Yes (because the item just added might not appear at the end).

The first item is number 0.

Example The following A-pex3 programming example adds the text contained in the variable named usertext to the Listbox with IDNumber 1, then highlights that item:

```
Listbox(1).AddItem = usertext
.LookAt = .LastAdded
.Tagged = -1
```

Also see AddItem, LookAt

Layout Object

Description The Layout object controls the window in which other objects are displayed.

Attributes
AutoCenter
AutoRedraw
AutoResize
BackColor
BackUpStack
BgndPicture
Bottom
Caption
ClickEvent
CloseEvent
Comment
Condition
ControlBox
DragDropEvent
Enabled
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
ForeColor
hDC
Height
HValue
hWnd
Icon
Left
LockUpdate
MaxButton
MenuStack
MinButton
MouseLeaveEvent
MouseOverEvent
Move
MoveEvent
Name
PopupMenu
Relocate
ResizeEvent
Right
SaveAsObject
Scrollable
SpecialEffect
SystemModal
Tile
TitleBar

Top
Update
VirtualHeight
VirtualWidth
Visible
VValue
Width
WindowBorder
WindowLayer
WindowState
ZOrder

Examples

Even though a Layout object does not have an IDNumber attribute, you can access it in A-pex3 programming as if the IDNumber were 1.

```
Layout(1).ZOrder = 0
```

Certain attributes can also be accessed directly via the Window "object."

```
Window(0).ZOrder = 0          $$ (0) is current window
```

Left Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects

Description — Controls the horizontal display location of the object.

Details — Specify in units of pixels. By default, the top edge of the window is 0.

The easiest way to adjust the Left attribute of an object at design time is via the VisualPage editor. First, click on the object to focus on it, then point to one of the sizing handles and either 1) press and hold down the left mouse button, and drag the mouse to move an edge, or 2) press and hold down the right mouse button, and drag the mouse to move the whole object.

For windows, the Left attribute is controlled via the Layout object. It specifies the distance from the left edge of the screen.

Example — The following A-pex3 program centers window number 1 on the screen.

```
dwidth = gsm(0)                $$ width of display
wwidth = Window(1).Width$$ width of window
Window(1).Left = (dwidth - wwidth) \ 2
```

Also see — AutoCenter, Height, Move, Right, Top, Width

Len() Function

Applies to A-pex3 programming

Description Returns the length of (number of characters in) a character string.

Syntax len(Which)

Example The following example sums the length of the strings stored in the array named files:

```
ptr = arr("files")$$ get number of elements
chars = 0               $$ initialize
DO IF ptr >= 0
  chars = chars + len(files(ptr))
  ptr--
LOOP
```

LetterRotation Attribute

Applies to	Frame object
Description	Specifies the angle with which the individual letters of the Caption are drawn.
Settings	0 to 360 (degrees)
Details	Several caveats apply here. Only vector fonts can be rotated. If the letters do not rotate, it is most likely because the font specified by FontName is not a vector font. The following vector fonts are supplied with Windows 3.1: Modern, Roman and Script. Even for vector fonts, Windows seems to have trouble rotating smoothly through the 0 to 360 degree range of this attribute. You will need to experiment to find the best appearance. Use at your own risk.
Also see	CaptionRotation, FontName

LightColor Attribute

Applies to	Button, Check, Combo, Frame, Gauge, Listbox, Option objects
Description	Specifies the color to employ as the bright color for 3-D shadowing effects.
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.
Also see	ShadowColor

LINE Command

Applies to A-pex3 Xgraphics programming

Description Draws a line.

Syntax LINE (X1, Y1, X2, Y2 [, Color])
or
LINE (X1, Y1 [,,,Color])

Details The LINE command draws a line from X1, Y1 to X2, Y2. It employs the current pen settings as determined by previous STYLE commands.

If you omit X2 and Y2, LINE draws a line from the previous drawing location to X1, Y1.

If you omit the Color parameter, Everest employs the current foreground color as set via a previous COLOR command.

Example The following example disables normal event handling and draws a line to the location pointed to each time the user clicks a mouse button. To run this example, place a Layout object at the top of the page, and set its ClickEvent to -200. Then, add a Program object and place the following code in it:

```
prevx = 0: prevy = 0    $$ init vars
dummyvar = key(1)       $$ disable normal events
STYLE (2, 6)            $$ invert color
DO
  event = key(3)        $$ wait for next event
  IF event > 0 THEN     $$ user pressed a key
    OUTLOOP             $$ exit loop
  ELSEIF event = -200 THEN
    LINE (prevx, prevy, sysvar(9), sysvar(10))
    prevx = sysvar(9)   $$ remember last x
    prevy = sysvar(10)  $$ remember last y
  ENDIF
LOOP
dummyvar = key(0)       $$ enable normal events
```

Also see Line Object, STYLE

Line Object

Description The Line object draws a line on the page.

Attributes
BorderColor
BorderWidth
Comment
Condition
Create
Destroy
DrawMode
IDNumber
Initially
Name
OutlineStyle
Visible
X1
X2
Y1
Y2

Details Use a Line object when you want to quickly and easily draw a line on the window. Because you can alter its attributes, a Line object is also handy for lines that need to change color, location, etc. at run time.

If you have many lines to draw from one page, you should employ the A-pex3 Xgraphics Line command instead because Windows handles it more efficiently.

Everest draws Line objects in a layer beneath other objects, but above Xgraphics vector graphics and the BgndPicture image.

Due to a bug in Windows, the Line object erases previously drawn Xgraphics located within a rectangular area bounded by the Line. If this creates a problem for your project, employ the LINE command instead.

Also see BgndPicture, LINE, Shape

ListBox Object

Description The ListBox object displays a group of items in one or more columns and allows the user to select one or more of them.

Attributes
AddItem
Answers1
Answers2
AntIncorrect1
BorderColor
BorderType
Bottom
Caption
ClickEvent
ColChar
ColCount
Comment
Condition
Create
DblClickEvent
Destroy
Divider
DragMode
EdgeSizeInner
EdgeStyleInner
Enabled
FillColor
FindString
Font3d
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
FoundIndex
GotFocusEvent
Height
IDNumber
Ignore
Initially
Item
ItemAlignment
ItemColor
ItemCount
ItemIndex
ItemList
Judgment
LastAdded
Left
LightColor

LookAt
LostFocusEvent
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
Name
RemoveItem
Right
SaveAsObject
SetFocus
ShadowColor
Sorted
TabOrder
TabStop
Tagged
TaggedCount
TaggedList
TagStyle
Top
TopIndex
Update
Visible
Width
Zev
ZOrder

Details

Listbox items are often accessed at run time. To learn how to do so, refer to the AddItem, Item, LookAt and Tagged attributes.

Due to limitations in Windows, changing certain Listbox attributes, such as Sorted, can cause the list items and/or colors to be reset.

To create a multi-column Listbox, refer to ColCount.

Also see

Combo Object

Load File Window

Everest displays the Load File window when you double click on attributes such as BgndPicture and FileName. The Load File Window helps you find and external file to use in your project.

DRIVES and DIRECTORIES

Use the Drives and Directories dialog boxes to find files stored in other locations. When you change these settings, Everest updates the Files listbox to show the names of appropriate files.

FILES LISTBOX

To choose a file shown in the Files listbox (i.e. select it for use), double click on it. If you want to perform some action on the file, such as Peek, click on its name only once to highlight it.

PEEKING

Everest lets you view the contents of many types of files, such as .BMP, .PCX and .WAV. To view (or listen to) a file, highlight its name, and click the Peek button. This feature is very handy when you are unsure of the actual contents of a file.

SEARCH

If you know the name of a file (or a portion of the name), but don't know in which subdirectory it resides, click the Search button. Everest will ask what you want to search for. For example, to search for all .PCX files that start with the letter e on drive C:, you would enter `c:e*.pcx`. You can also search for more than one file type at a time. For example, to search for all .GIF and .JPG files on drive D:, you would enter `d:*.gif;*.jpg`.

The Search feature automatically scans all subdirectories (including nested ones) on the drive you specify. It is surprisingly fast, and very handy when you are hunting for a particular file. The results of the search are displayed in the "Search or Embedded Results" listbox.

COPY FILES

Many authors copy external files into the same subdirectory as the book. This helps to gather the files together for future editing. The Copy File feature is especially handy when, for example, you find a desired file on a clipart CD-ROM. You can easily copy it to the book's location, then use it from there. For more information on why this is a good approach, see File Handling.

EMBEDDED

As described in File Handling, you can embed files directly into the book. To view a list of the files previously embedded, click on the Embedded button. The results are displayed in the "Search or Embedded Results" listbox. To select an embedded file for use, simply double click on its name.

EMBED FILE

If you see an external file you would like to embed into the book, simply highlight its name, and click on Embed. To delete and/or update embedded files, employ the Embed Manager utility.

LockUpdate Attribute

Applies to Layout object

Description Controls how objects are visually removed and added to a window at run time. Write only.

Settings

Yes	hold pending visual changes until next Wait object
No	allow pending changes, and update window step-by-step
1	(run time only) same as Yes, but do not change mouse cursor

Details When LockUpdate is No, Everest displays objects as they are encountered in the page. This means the user can see objects being added and removed from the window step-by-step.

If you prefer that the objects and changes to the window appear all at once (rather than step-by-step), set LockUpdate to Yes. When LockUpdate is Yes, Everest holds the changes in memory, then updates the window all at once when the next Wait object is encountered in the page. Be sure to put a Wait object in the page; if no Wait object is encountered, the window might not be updated visually, or other unpredictable results may occur. While changes are pending, the mouse cursor appears as an hourglass.

You can also control LockUpdate in A-pex3 programming (thus eliminating the Wait object requirement). Use the Window "object" to refer to the LockUpdate attribute as in the following examples:

```
Window(0).LockUpdate = -1        $$ same as Yes

Window(0).LockUpdate = 0         $$ same as No

Window(0).LockUpdate = 1         $$ Yes, but no hourglass
```

The number inside the parentheses is the window number (0 to 8). Window number 0 refers to the current window.

Notes Only one window can be locked at a time. Do not open a new window (such as via BRANCH or CALL) or move the window while LockUpdate is enabled.

This feature employs the Windows API LockWindowUpdate function, and is subject to its limitations. Use LockUpdate at your own risk.

If A-pex3 Xgraphics commands do not plot when you enable LockUpdate, try also enabling AutoRedraw.

Also see DoEvents, Initially, Update

Lod() Function

Applies to A-pex3 programming

Description Copies a value from one element of an array into other elements of that array.

Syntax lod(Arrayname(Element) [, Separator])

Details Arrayname is the name of the array in which to copy the value. Element is the element of the array that already contains the value to copy. The Lod() function copies the value in Arrayname(Element) to all elements numbered from 1 to Element-1.

If you include the Separator parameter, Lod() parses the value in Arrayname(Element) using the Separator character as delimiter, and stores each parsed item into an array element, starting with element number 1.

The Lod() function is much faster than a DO...LOOP for initializing or clearing an array.

Example The following example erases the contents of elements 1 to 100 in the array named bigtext:

```
bigtext(100) = "" $$ init value to copy
dummyvar = lod(bigtext(100))
```

The following example copies the elements of the array named source into the array named dest:

```
elements = arr("source")
REDIM dest(elements)
dest(elements) = sum(source(elements), ";")
dummyvar = lod(dest(elements), ";")
```

Also see Srt() Function, Sum() Function

Log() Function

Applies to	A-pex3 programming
Description	Returns the natural logarithm of a number.
Syntax	log(Numeric)
Details	Numeric must be a number greater than 0.

LookAt Attribute

Applies to Combo, Listbox objects

Description Set to a numeric value that indicates which item in the list will be referenced in subsequent Item and Tagged attribute uses.

Details The first item in the list is number 0.

Example The following A-pex3 program reads the items from the Combo object with IDNumber 1 and concatenates them into a variable named all, separating each with a semicolon:

```
max = Combo(1).ItemCount
ptr = 0: all = ""
DO IF ptr < max
  .LookAt = ptr
  all = all + ";" + .Item
  ptr++
LOOP
```

Also see Item, ItemIndex, Pik() Function, Tagged, TopIndex

LOOP Command

Applies to	A-pex3 programming
Description	Marks the end of a DO...LOOP block.
Syntax	LOOP [IF <Condition>]
Details	Refer to the DO command.
Notes	If you want to use additional A-pex3 programming on the same line after a LOOP command, be sure to put a space and a colon (in that order) after the LOOP command.
Also see	RELOOP

LostFocusEvent Attribute

Applies to	Button, Check, Combo, HScroll, Input, Listbox, Mask, Media, OLE, Option, VScroll objects
Description	This event fires when the object gives up the focus (the highlight).
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	When you enter a number or string constant for LostFocusEvent, you are merely telling Everest what event to generate when the object loses the focus. To make use of that event (i.e. detect it and do something useful), you must include a Wait object in your page.
Also see	ClickEvent, GotFocusEvent, SetFocus

LPRINT Command

Applies to A-pex3 programming

Description Sends text to the active Windows printer.

Syntax LPRINT (Action [,Value1] [,Value2])

Details Use LPRINT to print text on the printer; choose a particular action via the Action parameter. Express Action with one of the following numbers:

1	print text specified by Value1
2	same as 1, except send an EndDocument code when done printing
0	set number of lines per page to Value1; Everest will automatically send a form feed code when the page is full; set to 0 for no automatic form feed
-1	send a form feed code
-2	send an EndDocument code (which releases the job for printing)
-3	move "print head" position (next printing location on this page) to a horizontal location of Value1 and/or a vertical location of Value2

Example The following example sets a 54-line page, prints the contents of Textbox(1), then releases the job for printing:

```
LPRINT (0, 54)
LPRINT (1, Textbox(1).Text)
LPRINT (-2)
```

Notes For proper operation, include a space between LPRINT and (.

To print the visual contents of a window, use Ext(19) or Ext(119).

Also see PRINT, Sysvar(21), Sysvar(22), Wrp() Function

Ltr() Function

Applies to	A-pex3 programming
Description	Returns the character string String with leading blank spaces (ASCII 32) removed.
Syntax	ltr(String)
Example	The following example removes the spaces (if any) from the beginning of the string stored in variable mytext: `mytext = ltr(mytext)`
Also see	Rpl() Function, Rtr() Function

Lwr() Function

Applies to	A-pex3 programming
Description	Returns the character string String with characters A to Z converted to the corresponding lower-case letters a to z. Or, returns Numeric rounded down to the nearest integer.
Syntax	lwr(String) lwr(Numeric)
Details	Note that the Lwr() function performs either one of two actions based on whether the parameter you pass is a character string or a number.
Example	The following A-pex3 example converts the contents of the Input object with IDNumber 1 to lower case:

```
Input(1).Text = lwr(Input(1).Text)
```

Also see Upr() Function, Val() Function

Mask Object

Description The Mask object is a single-line "fill-in-the-blank" style user interaction field ideal for validating specialized input.

Attributes
AdjustResponse
Answers1...Answers8
AntIncorrect1
AntIncorrect2
BackColor
Bottom
CMIData
Comment
Condition
Create
Destroy
DragMode
Enabled
EOFContinue
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
ForeColor
Format
GotFocusEvent
Height
IDNumber
Ignore
Initially
InputTemplate
InvalidEvent
JudgeVar
Judgment
Left
LostFocusEvent
MaxLength
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
Name
Preset
PromptChar
ResponseVar
Right
SaveAsObject
SetFocus
TabOrder

TabStop
Text
Top
Tries
Update
Visible
Width
Zev
ZOrder

Details

Employ a Mask object to obtain a specially formatted response from the user. For example, the Mask object is ideal for obtaining a telephone number, social security number, or other response that has a particular form. You specify the form via the InputTemplate attribute.

If the user types a character that is outside the range allowed by the InputTemplate, the Mask object generates the InvalidEvent event code. You can trap the event via a Wait object and take any desired action, such as displaying a warning message.

The user's response is returned in the variable whose name you enter for the ResponseVar attribute.

To judge the user's response for accuracy, place answer specifications in the Answers attributes. The answer judgment is returned in the variable whose name you enter for the JudgeVar attribute.

Also see

Answers, Input Object, JudgeVar, ResponseVar

Max Attribute

Applies to	Gauge, HScroll, VScroll objects
Description	Sets the value of the upper bound of the object. For a Gauge, Max should always be set to a value greater than that of Min.
Also see	FillValue, Min, Value

MaxButton Attribute

Applies to	Layout object	
Description	Determines whether a maximize button is displayed in the TitleBar of the window.	
Settings	Yes	display maximize button
	No	do not display maximize button
Also see	ControlBox, MinButton	

MaxDrop Attribute

Applies to	Combo object
Description	Sets the number of lines of items to display in the drop-down area of the Combo object.
Notes	Due to a bug in the MicroHelp control that drives this object, this attribute might not produce the correct number of lines.
Also see	Style

MaxLength Attribute

Applies to	Mask object
Description	Sets the maximum number of characters a user can enter in the editing area.
Settings	1 to 64
Also see	EOFContinue, InputTemplate

Mbx() Function

Applies to A-pex3 programming

Description Displays a message window and waits for the user to choose a button. Returns a number that indicates which button the user chose.

Syntax mbx(Message [, Type [, Caption]] [, Beep])

Message is a character string that you want to display in the window.

Type is a number that indicates the kind of buttons and icon to display in the window.

Caption is a character string to display as the title bar of the window.

Beep is a number that specifies a sound effect to play.

Details This function is very handy when you want to display a brief message (such as an error description) to the user.

TYPE PARAMETER

To specify the kind of button(s) to display in the window, use one of the following values for Type:

0	OK
1	OK, Cancel
2	Abort, Retry, Ignore
3	Yes, No, Cancel
4	Yes, No
5	Retry, Cancel

ICONS

To include an icon in the window, add one of the following values to Type:

16	Stop
32	Question mark
48	Exclamation point
64	Information i

DEFAULT BUTTON

To assign a default button, add one of the following values to Type:

0	First button is the default
256	Second button is the default
512	Third button is the default

The default button is that chosen if the user simply presses Enter.

MODAL WINDOW

To make the message box system modal (i.e. ignore user mouse clicks outside), add the following value to Type:

4096 Box is system modal (might hide icon due to Windows bug)

BEEP PARAMETER

Include the optional fourth parameter to play an alert sound when the message box appears. Use one of the following values.

-1 Standard beep through the computer's speaker
0 SystemDefault sound (as defined in WIN.INI)
16 SystemHand (Stop) sound (as defined in WIN.INI)
32 SystemQuestion sound (as defined in WIN.INI)
48 SystemExclamation sound (as defined in WIN.INI)
64 SystemAsterisk sound (as defined in WIN.INI)

RETURNED VALUES

The Mbx() function returns one of the following values based on the button the user chooses:

1 OK
2 Cancel
3 Abort
4 Retry
5 Ignore
6 Yes
7 No

Example The following A-pex3 programming example displays a message window containing a question mark icon, Yes/No/Cancel buttons, and a message that asks whether the user wants to save changes:

```
choice = mbx("Save changes?", 35)
IF choice = 6 THEN            $$ yes (save)
  answer = Textbox(1).Text
ELSEIF choice = 7 THEN        $$ no
  answer = ""
ELSEIF choice = 2 THEN        $$ cancel (redo)
  GOTO top                    $$ LABEL top is elsewhere
ENDIF
```

Also see Ibx() Function, SystemModal

Mci() Function

Applies to A-pex3 programming

Description Communicates directly with the Windows Media Control Interface.

Syntax mci(Operation, CommandString [, ReturnLength])

Operation is a number that indicates the action to perform.

CommandString is a character string that contains the instructions to send the Media Control Interface.

ReturnLength is a number that specifies the length of a string buffer for information to be returned by the Media Control Interface. Needed only when Action is 1.

Details The Mci() function sends commands to the Windows Media Control Interface (MCI) to directly operate multimedia devices. Typical CommandStrings resemble "Play" and "Pause." The Mci() function performs various actions based on the value of the Operation parameter:

Operation	Action
0	Sends CommandString to the Windows API MCIExecute function. Returns a numeric error code (0 means no error).
1	Sends CommandString to the Windows API MCISendString function. Returns information of length ReturnLength.
2	Calls the Windows API MCIGetErrorString function. Returns the error message that corresponds to the number specified in CommandString.
3	Plays the .WAV file whose filename is specified in CommandString. Returns a numeric error code (0 means no error).

When you expect the MCI to return information, set Operation to 1, and include a number for the ReturnLength parameter. This tells Everest how long a buffer (how many characters) to create to hold the returned information. This buffer is returned by the Mci() function. Numeric error codes are returns in the Sysvar(1) variable.

To learn what CommandStrings are accepted by a multimedia device, consult the documentation for that device, or a Microsoft Windows Multimedia Programmer's Guide.

Before exiting your project, be sure to close any devices you open via the Mci() function.

Examples The following example plays back the Microsoft Video for Windows COWBOY.AVI file, and scales the image to fit the size of the picture box. To try this example, create a new page, add a Picture object with IDNumber 1, and a Program object that contains:

```
z = mci(0, "open cowboy.avi alias anyname type AVIVideo")
z = mci(0, "window anyname handle " + picture(1).hWnd)
```

```
size = picture(1).width $+ " " $+ picture(1).height
z = mci(0, "put anyname destination at 0 0 " + size)
z = mci(0, "play anyname wait")
z = mci(0, "close anyname")
```

For reliable use, you should add error checking to the example above.

The following example plays a multimedia element that was initially opened via Everest's Media object (with IDNumber 1):

```
z = "play " + Media(1).FileName + " wait"
z = mci(0, z)
```

The following example plays the TADA.WAV file after checking if the computer is capable of it:

```
IF ext(114) > 0 THEN
  z = mci(3, "c:\windows\tada.wav")
ELSE
  z = mbx("Your computer has no support for .WAV audio.")
ENDIF
```

Also see Media Object

Media Object

Description The Media object is used to operate multimedia devices such as CD players, videodiscs, sound boards, digitized video, etc.

Attributes
AutoScale
Bottom
Command
Comment
Create
Destroy
DeviceType
DisplayIn
DoneEvent
DragMode
Enabled
EndAt
FileName
GotFocusEvent
Height
IDNumber
Left
LostFocusEvent
MouseLeaveEvent
MouseOverEvent
MousePointer
Name
Orientation
Position
Right
SaveAsObject
ShowButtons
Silent
StartAt
TabOrder
TimeFormat
Top
UpdateEvent
UpdateInterval
Wait
Width
Zev
ZOrder

Details Via the Media object, you can easily incorporate Multimedia in your project.

Before setting other attributes, be sure to choose a DeviceType. Most DeviceTypes require that you have previously installed an appropriate software driver via the Windows Control Panel. These drivers are supplied by hardware manufacturers.

A given page can have more than one Media object. If you want all the Media objects

on a page to control the same DeviceType, set their IDNumber attributes to the same number.

If you prefer to control the Multimedia devices yourself via programming code, employ the Mci() Function instead of, or in cooperation with, the Media object. The Mci() Function can operate independently of the Media object.

Microsoft is aware of a bug in Windows that, during editing, sometimes causes the VisualPage editor's sizing handles to get stuck on an object or disappear from a page that contains a Media object. To work around this problem, try clicking on a different object in the Book Editor.

Also see Animate Object, Mci() Function

Menu Object

Description Use the Menu object to add or modify pull-down menus in the page window.

Attributes
Comment
Condition
IDNumber
Name
NewMenu
SaveAsObject

Details To open the Menu Editor window, double click on the Menu object icon in the Book Editor.

The menu editor is a free-form text editor. Here you enter the text and information about the pull-down menu items. Each line of text you enter corresponds to a menu item. The syntax is:

ItemCaption, EventCode, [Enabled], [Checked] [, Column, Row]

where

ItemCaption	= text to display
EventCode	= code to generate when user selects item
Enabled	0 = gray the ItemCaption (disabled)
	1 = activate the item (default)
Checked	0 = no check mark (default)
	1 = display check mark to left of ItemCaption
Column	# = column number of item to update
Row	# = row number of item to update

Examples You might enter something like:

```
File, 1
  New, 4078
  Load, 4076
  Save, 4083
  -
  Quit, 4081
Edit, 2
  Cut, 2088
  Copy, 2067
  Paste, 2086
```

The first number after the pull-down item's text caption is the event code you wish to generate when the user selects the item. You can invent any numeric code you want (between -32000 and 32000). Typically, you include a Wait object in your script to trap the event codes and take specific actions.

Sometimes it is useful to set your event codes to match those of keys the user can press. In the example above, event code 4078 matches the code generated if the user presses <Alt-n>. Keycodes can be found in Appendix A.

Everest places non-indented items along the top of the menu (maximum of 8 columns). Items you indent (with at least one space) appear in the pull-down boxes beneath (maximum of 20 rows).

Use a hyphen (-) to put a divider line in the menu.

UPDATING MENUS

You can update an existing menu item to, for example, put a check mark next to it. To update an item, include the Column and Row parameters. You can omit the other parameters to leave them unchanged. For example:

,,,1,2,3

puts a check mark next to the item in column 2, row 3. Also, be sure to set the NewMenu attribute of the Menu object to No.

SHORTCUT KEYS

You cannot designate/display a shortcut key for a menu item. However, you can include the name of a desired shortcut key in the ItemCaption, and set the EventCode to match that generated by the key. For example:

```
Help       F1
```

There is no automatic way to right-justify the shortcut key's name (you can pad with spaces manually).

ACCESS KEYS

Even though you cannot designate a shortcut key, you can specify an access key. Prefix the desired access key letter with an ampersand (&). For example:

```
&Quit
```

makes Q the access key. A user can tap the Alt key, followed by an access key to activate its feature.

Notes

The Enabled and Checked features are available only for menu items that appear in the drop-down portion of the menu (i.e. those lines indented in the Menu Editor).

The color used for menu items is determined by the Windows system color settings.

Due to a bug in Windows, we do not recommend using a Menu object when the TitleBar attribute of the Layout object is set to No.

Also see

NewMenu, PopupMenu

MenuAction Attribute

Applies to	Wait object
Description	Specifies the action to perform when the MenuActivator event is triggered.
Double click	First: sets MenuAction to BRANCH @menu. Next: Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.
Details	Enter any single line of A-pex3 programming code. When a Wait object sees that an event code matches the MenuActivator event, it traps that event code, and performs the MenuAction. Most authors employ the MenuActivator and MenuAction to trap a user's request to branch to a menu.
Example	To branch to the most recent page placed on the menu stack, enter the following for MenuAction: `BRANCH @menu`
Also see	MenuActivator, MenuStack

MenuActivator Attribute

Applies to	Wait object
Description	Specifies the numeric event code that triggers the MenuAction.
Settings	-32000 to 32000, or a string surrounded by quotes
Double click	Opens event code dialog box. Press the desired key to automatically generate the corresponding event code.
Details	Everest watches the events that occur in your project, and checks if one matches the event code you specify as the MenuActivator. If a match is found, the event is removed from the queue, and Everest performs the MenuAction. Most authors employ the MenuActivator to detect when a user has pressed the "back up to previous menu page" key.
Example	To make a Ctrl+M keypress the event that invokes the MenuAction, set the MenuActivator to the event code for Ctrl+M: 2077.
Also see	MenuAction, Wait Object

MenuStack Attribute

Applies to Layout object

Description Controls whether Everest places the current page on the menu stack.

Details Many authors find it convenient to divide a project into sections, and provide menus to access the sections. Often a project has several layers of menus.

When a user wishes to return to a menu, usually he wants to return to the one most recently viewed. By setting the MenuStack attribute, you can tell Everest to keep track of the names of the menu pages encountered by the user. Everest stores this list in Sysvar(81) to Sysvar(88).

Later, to branch back to the most recent menu, use the A-pex3 command

```
BRANCH @menu
```

Also see BackUpStack, MenuAction, MenuActivator

Mid() Function

Applies to A-pex3 programming

Description Returns a portion of a string of characters.

Syntax mid(String, Start [, Length])

String is the character string to copy from.

Start is the starting location in String. The leftmost character is 1.

Length is an optional parameter that specifies the number of characters of String to return.

Details Use Mid() to make a new string from a portion of an existing one, that is, copy a portion of a string.

If Start+Length is greater than the length of String, or Length is omitted, the whole String is returned, beginning at Start.

Example The following program uses Mid() to copy a portion of a string.

```
fullname = "Last, First"
ptr = fullname * ","          $$ find comma
IF ptr = 0 THEN
  firstname = ""
ELSE
  firstname = Ltr(Mid(fullname, ptr+1))
ENDIF
```

Notes To copy a single character from the middle of a String, use the ^^ operator (it is faster than the Mid() function).

`Mid(String, Start, Length)` returns the same string as the expression `String - Start \ Length`.

Also see Len() Function, Pik() Function

Min Attribute

Applies to	Gauge, HScroll, VScroll objects
Description	Sets the value of the lower bound of the object. For a Gauge, Min should always be set to a value less than that of Max.
Also see	FillValue, Max, Value

MinButton Attribute

Applies to	Layout object
Description	Determines whether a minimize button is displayed in the TitleBar of the window.
Settings	Yes display minimize button No do not display minimize button
Also see	ControlBox, MaxButton

Mki() Function

Applies to A-pex3 programming

Description Returns an integer in a two-character string representation.

Details The Mki() function is handy for packing a group of numeric values (in the range 32,767 to 32,767) into a string. Since each value is represented by 2 characters, individual values in a string can be easily extracted via parsing.

Use the Cvi() function to convert back to an integer.

Example The following example generates 10 random numbers between 1 and 1000 and stores them all in one 20-byte string:

```
count = 1: packed = ""
DO
  packed = packed + mki(rnd(1000))
  count++
LOOP IF count <= 10
```

Also see Chr() Function, Cvi() Function

MouseLeaveEvent Attribute

Applies to	Animate, Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox objects
Description	This event fires when the mouse cursor moves off an object (and onto another in the list above).
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Example	The following example for a Button with Name QuitButton changes its background color to gray when the mouse cursor exits it: `QuitButton.FillColor = rgb(128, 128, 128)`
Notes	MouseLeaveEvents are generated only for the active wiNdow and its objects.
Also see	MouseOverEvent, Mse() Function

MouseOverEvent Attribute

Applies to	Animate, Button, Check, Combo, Flextext, Frame, Gauge, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox objects
Description	This event fires when the mouse cursor enters an object after leaving another (in the list above).
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Example	The following example for a Button updates the contents of the Textbox with IDNumber 1 when the cursor enters the Button: `Textbox(1).Text = "Your mouse now points to a button."`
Notes	The MouseOverEvent for a Shape object has the highest priority (i.e. even if another object is on top of the Shape thereby obscuring it, Everest still fires the MouseOverEvent for the Shape). MouseOverEvents are generated only for the active window and its objects.
Also see	MouseLeaveEvent, Mse() Function

MousePointer Attribute

Applies to — Animate, Button, Check, Combo, Frame, Gauge, HScroll, Input, Listbox, Mask, Media, Option, Picture, SPicture, Textbox, VScroll objects

Description — Controls the appearance of the mouse cursor while the cursor is positioned over the object. Accessible only at run time via A-pex3 programming.

Settings

0	default
1	arrow
2	cross-hairs
3	I-beam
4	icon
5	N, S, E, W arrows
6	NE, SW arrows
7	N, S arrows
8	NW, SE arrows
9	W, E arrows
10	up arrow
11	hourglass
12	"not allowed" symbol

Example — The following example tells Everest to display a cross-hair style mouse cursor when the user positions it over the Picture box with IDNumber 1:

```
Picture(1).MousePointer = 2
```

Also see — Mse() Function, NormalPointer, ShapePointer

MouseStayEvent Attribute

Applies to	Shape object
Description	This event fires if the mouse cursor remains over the Shape for a period of time.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	Most authors use this feature to create so-called "balloon help" in a project. Balloon help is a help message, picture or whatever you want, that appears after the user has kept the mouse cursor within a (typically small) area for a (typically short) duration of time. The MouseStayEvent fires if the user keeps the mouse cursor over the Shape for one second. If you want a different time duration, set Sysvar(173) equal to the number of seconds you prefer.
Example	The following example for MouseStayEvent makes the Picture with IDNumber 1 visible after the mouse cursor remains over the Shape: `Picture(1).Visible = -1`
Notes	The MouseStayEvent for a Shape object has the highest priority (i.e. even if another object is on top of the Shape thereby obscuring it, Everest can still fire the MouseStayEvent for the Shape). If you want to cancel a pending MouseStayEvent, for example after another event occurs, set Sysvar(174) to 1 as part of the processing of that other event. MouseStayEvents are generated only for the active window.
Also see	MouseOverEvent, Mse() Function

Move Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects

Description — Relocates and resizes an object in one step. Also, returns a string that contains the Left, Top, Width and Height attributes of an object.

Details — The Move attribute can change the location and size of an object in one step. The end result is the same as setting the Left, Top, Width and Height attributes individually. With Move, all attributes change at once, reducing the amount of replot flicker on the screen.

Set Move equal to a string containing either one or two pairs of X-Y coordinates separated with commas. The first pair sets the Left and Top attributes. The second (optional) pair sets the Width and Height.

The Reg() function is helpful for converting coordinate pair values that are in variables into a string that Move accepts.

Examples — The following example moves the Picture with IDNumber 1 to the location 50, 60 and sets the Width and Height to 70 and 80 respectively:

```
Picture(1).Move = "50,60,70,80"
```

The following example moves the Textbox with IDNumber 1 to a location specified via variables:

```
Textbox(1).Move = reg(newleft, newtop)
```

The following example relocates and resizes the Button with IDNumber 1 in window number 2 to match that of the Button with IDNumber 1 in window number 1:

```
Window(2)!Button(1).Move = Window(1)!Button(1).Move
```

Also see — Height, Left, Reg() Function, Top, Width

MoveEvent Attribute

Applies to	Layout object
Description	Event code to generate, or programming to perform, when the user moves the window.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	When you enter a number or string constant for MoveEvent, you are merely telling Everest what event to generate when the user changes the location of the window (i.e. when the Top and/or Left attributes change due to user action). To make use of that event (i.e. detect it and do something useful, like rearrange objects in the window), you must include a Wait object in your page.
Notes	As the corresponding action for a MoveEvent, do not close the window.
Also see	CloseEvent, Relocate, ResizeEvent

Mse() Function

Applies to A-pex3 programming

Description Controls the appearance and location of the mouse cursor.

Syntax mse(Operation [, X [, Y] [, R, B])

Operation is a number that specifies the action to perform.

X is a number used by certain Operations.

Y is a number that represents a vertical page location used by certain Operations.

R is a number that represents the right edge of the area for Operation -2

B is a number that represents the bottom edge of the area for Operation -2

Details The Mse() function performs various actions based on the Operation you employ:

-2 restrict mouse pointer to area bounded by X, Y, R, B; to restore unrestricted mouse movement, omit X,Y,R,B parameters.

-1 mouse check; returns 0 if no mouse is installed, a non-zero value if one is found

0 sets mouse cursor appearance; changes the shape of the mouse cursor for the whole screen; specify one of the following values in X:

0	default
1	arrow
2	cross-hairs
3	I-beam
4	icon
5	N, S, E, W arrows
6	NE, SW arrows
7	N, S arrows
8	NW, SE arrows
9	W, E arrows
10	up arrow
11	hourglass
12	"not allowed" symbol

1 returns horizontal location of mouse cursor relative to the left edge of the window

2 returns vertical location of mouse cursor relative to the top of the window

3 returns horizontal location of mouse cursor relative to the left edge of the display

4 returns vertical location of mouse cursor relative to the top of the display

5 sets horizontal and vertical location of the mouse cursor to X and Y respectively,

relative to the upper-left corner of the window

6 sets horizontal and vertical location of the mouse cursor to X and Y respectively, relative to the upper-left corner of the display

7 returns last MouseOverEvent object number

8 returns last MouseOverEvent IDNumber

15 same as 5, except animates cursor movement

16 same as 6, except animates cursor movement

Examples The following A-pex3 calculation moves the mouse cursor to the upper-left corner of the window:

```
dummyvar = mse(5, 0, 0)
```

The following A-pex3 calculation animates the mouse cursor onto the Button object with IDNumber 1:

```
dummyvar = mse(15, Button(1).Left+10, Button(1).Top+10)
```

The following A-pex3 calculation restricts the mouse cursor to movement within the current window:

```
dummyvar = mse(-2, window(0).left, .top, .right, .bottom)
```

Also see MouseOverEvent, MousePointer, Reg() Function, Sysvar(9) Variable, Sysvar(10) Variable

Msg() Function

Applies to A-pex3 programming

Description Returns a short text message read from the EVEREST.MSG file.

Syntax msg(Numeric)

Details Everest keeps descriptive error messages in the EVEREST.MSG file. When an error occurs while you are authoring, Everest retrieves an explanation of the error from the file, and displays it in the error window.

Each message is represented by a number from 999 to 9999. To retrieve a particular message, pass the Msg() function its number.

Everest also stores other (non-error) messages in the file, such as default user log-on text. If you wish, you can modify the message file, or add additional messages to it. This ability is especially useful to authors creating projects in a foreign language.

The contents of the EVEREST.MSG file are stored in ASCII format. You can modify the file with a text editor, such as the DOS EDIT program, or the Windows Notepad. Keep each message on one line. If you add a message, assign it a number from 500 to 999. Be sure to keep the messages in numeric order (999 would be first).

Example The following A-pex3 example retrieves message -600 from the EVEREST.MSG file and displays it in the Textbox with IDNumber 1:

```
Textbox(1).Text = msg(-600)
```

MultiLine Attribute

Applies to	Button, Check, Frame, Input, Option, Textbox objects
Description	Controls whether text that is too long to fit horizontally within the object is automatically wrapped to the next line.
Settings	Yes allow long text to wrap No keep text on one line
Notes	When MultiLine is enabled, text begins plotting near the top of the object. To center text vertically on the object (except for Textboxes), set MultiLine to No. When MultiLine is enabled, you can override automatic wrapping and force Caption text to wrap by placing a Carriage Return character at the desired location(s). Press Alt+Enter while editing the Caption to insert a Carriage Return.
Also see	Alignment, Caption, ScrollBars

Name Attribute

Applies to All objects

Description Specifies the name that identifies an object. Available at design time only.

Settings A text string up to 19 alphanumeric characters (plus underlines and periods, but not spaces) in length. The Name must start with a letter from A to Z.

Details Everest assigns a default Name when you create an object. Most of the time, an object's Name has little significance. You can change the Name as you see fit to make it more meaningful. For example, when you drag and drop a new Button on the VisualPage editor, Everest might name it "page10_button_C" but you can change that to something like "quit_button" to make it more meaningful. To change the name, first highlight the object in the Book Editor, then, from the Edit menu, choose Rename.

There are a few situations in which the Name becomes significant:

1) if you enable SaveAsObject, Everest saves a copy of the object independently of the page; the object's Name is used to uniquely identify it.

2) the Name of a JLabel object can be used as the destination of a JUMP command.

3) a Name devoid of periods and spaces can be used in A-pex3 programming to refer to the object.

If you change the Name to match that of an object stored independently in the current book (i.e. SaveAsObject was enabled for it), Everest makes the object an instance of the one in the book.

Even though Everest lets you change the Name to match that of an object in a different class (for example, name a Textbox the same as a Picture), it is recommended that you avoid doing so to help minimize confusion.

Case is insignificant in the Name. (Everest internally stores the Name using lower-case letters.)

Example If the Name is unique in the page, and it contains no periods or spaces, you can use it in A-pex3 programming *within the same page* to refer to the object. For example, if you change the Name of the Button object with IDNumber 1 from, say, page1_button_A to MenuButton, you can refer to it in subsequent A-pex3 programming in either of the following ways:

```
Button(1).Enabled = -1

MenuButton.Enabled = -1
```

Do not use the Name to, at run time, change the value of attributes of those objects without IDNumbers.

Notes The Name of an object is independent of the name of the page. When you add a new

object to a page, Everest incorporates the page name in the object name simply to help find a unique Name. You can change the Name as you see fit.

Also see IDNumber

NewMenu Attribute

Applies to	Menu object	
Description	Determines whether the menu instructions should modify an existing menu in the window, or replace it entirely.	
Settings	Yes	replace old menu (if any)
	No	modify the old menu (if any)

NextAction Attribute

Applies to Wait object

Description Specifies the action to perform when a NextActivator event is triggered.

Double click Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.

Details Enter any single line of A-pex3 programming code. A commonly used NextAction is

```
BRANCH @next
```

which branches to the next page in the book.

When a Wait object sees that an event code matches the NextActivator event, it traps that event code, and performs the NextAction.

Most authors employ a NextActivator and NextAction to trap a user's request to branch to the next page.

If you simply want Everest to continue processing objects in the page that follow the Wait object, specify one or more NextActivators and leave NextAction empty. If you want to perform a command, and then continue, use the JUMP command as in the example below.

Example The following NextAction example causes project execution to continue with the page named page2:

```
BRANCH page2
```

The following NextAction example GOSUBs to the Program object named "routine", then tells Everest to continue processing objects in the page that follow the Wait:

```
GOSUB routine: JUMP @proceed
```

Notes To enhance run time performance, if NextAction is a BRANCH to another page in the same book, Everest preloads that page from disk while the user is viewing the current page. When possible, Everest also automatically prepares objects from the preloaded page, keeping them hidden until the user proceeds ahead to that page. Such preloaded objects display much faster than non-preloaded objects.

Even if a particular Wait object does not employ a NextActivator, you can still take advantage of the significant execution speed increase provided by Everest's automatic preloading feature. Simply place a "dummy" BRANCH command in NextAction...a BRANCH command that points to the page to which the user is most likely to proceed. Regardless of how the user branches to this page (via an OtherAction, via A-pex3 programming, etc.) Everest still makes use of the preloaded information.

Also see NextActivator

NextActivator Attribute

Applies to	Wait object
Description	Specifies the numeric event code that triggers the NextAction.
Settings	-32000 to 32000, or a string surrounded by quotes
Double click	Opens event code dialog box. Press the desired key to automatically generate the corresponding event code.
Details	Most authors employ the NextActivator to specify what event(s), such as Button ClickEvents, etc., will trigger branching to the next page via NextAction. Everest watches the events that occur in your project, and checks if one matches the event code you specify in NextActivator. If a match is found, the event is removed from the queue, and Everest performs the NextAction. If NextAction is empty, Everest continues to process the page with the object that follows the Wait. On pages with question fields, answer judging is NOT performed if the NextAction is triggered. To specify the event that triggers answer judging, use JudgeActivator.
Example	To specify a keypress as a NextActivator, enter the numeric keypress event code, or to have Everest generate the numeric code for you, double click on the NextActivator attribute, then press the desired key. The following entry for NextActivator watches for the PgDn key numeric event code, or the event string "next clicked" `34,"next clicked"`
Notes	Up to 16 events can be specified in NextActivator; separate the events with commas.
Also see	JudgeActivator, NextAction

NormalPointer Attribute

Applies to Flextext object

Description Controls the appearance of the mouse cursor while the cursor is positioned over the object, but not over a jump or popup word. Accessible only at run time via A-pex3 programming.

Settings

0	default
1	arrow
2	cross-hairs
3	I-beam
4	icon
5	N, S, E, W arrows
6	NE, SW arrows
7	N, S arrows
8	NW, SE arrows
9	W, E arrows
10	up arrow
11	hourglass
12	"not allowed" symbol
13	hand
14	arm
15	target
16	wand
17	boy
18	girl
19	pencil
20	lightning
21	key
22	telephone
23	question mark

Example The following example tells Everest to display a cross-hair style mouse cursor when the user positions it over the Flextext object with IDNumber 1:

```
Flextext(1).NormalPointer = 2
```

Also see JumpPointer, MousePointer, PopupPointer

Obj() Function

Applies to A-pex3 programming

Description Returns information about the objects in a window. Recommended for use by experienced programmers only.

Syntax obj(Operation [, ClassName] [, IDNumber] [, Window] [, Attribute] [,Value])

Operation is a number that specifies the action to perform.

ClassName is a character string that contains the name of an object, such as Textbox.

IDNumber is the identification number attribute for the ClassName object.

Window is an optional value from 0 to 8, or -2, that specifies the number of the window to examine. When omitted or 0, Everest assumes the current window.

Attribute is the name of the object attribute to read or set via Operation 6 or 7.

Value is the value to assign Attribute when Operation is 7.

Details For specialized applications, it may be useful for your A-pex3 program to know whether certain objects already exist within a window, or which have the focus. For example, that information could be used by the program to determine where to JUMP in a page.

Use one of the following Operations:

0 returns a number greater than 0 if the ClassName object with IDNumber exists in Window.

1 returns a string of bytes, one byte for each object in ClassName. Use the Asc() function to convert each byte to a number; the number is the IDNumber of the object.

2 returns a number that indicates the last object to receive focus in any window. See the table below to relate the number to an object.

3 returns a number that is the IDNumber of the last object to receive focus in any window. Also see GotFocusEvent attribute.

4 returns the object number that corresponds to ClassName.

5 returns the Class Name for the object number specified in the ClassName parameter.

6 returns the value of the Attribute specified. Places an error code (if any) in Sysvar(1), or -1 if no error.

7 sets the value of the Attribute specified to Value. Returns an error code (if any) or -1 if no error.

Table for Obj(2), Obj(4) and Obj(5) Function:

Object NumberName	Class
50	Layout (the window)
65	Animate
66	PicBin
67	Program
70	Frame
71	Gauge
72	HyperHlp
73	Line
74	Judge
75	Timer
76	JLabel
77	Media
78	Flextext
79	OLE
80	SPicture
83	Special
84	Textbox
85	Erase
86	Picture
87	Wait
88	Include
90	Shape
98	Button
99	Check
101	Mask
104	HScroll
105	Input
107	Combo
108	Listbox
109	Menu
111	Option
118	VScroll
>127	External object

Example

The following A-pex3 program determines and displays the number of Media objects in the current window, as well as their IDNumbers:

```
mlist = obj(1, "Media")
ptr = 1: idlist = ""
DO IF ptr <= len(mlist)
  idlist = idlist + " " $+ asc(mlist ^^ ptr)
  ptr++                 $$ faster than ptr = ptr + 1
LOOP
message = "There are " + len(mlist)
message = message + " Media objects, with ID#s "
message = message + idlist
```

```
dummyvar = mbx(message, 64)
```

Notes

When viewing a page in via single-page Preview, the default Window is -2.

The Sysvar(8) variable contains the number of the active window.

For easy reference, the Class name of an object is displayed at the top of the ToolSet window when you click once on a ToolSet icon.

Also see

Create, Destroy, IDNumber, SetFocus

Object Manager Window

The Object Manager Window via the main Author window's Utilities pull-down menu. The Object Manager helps you keep track of master objects...those with SaveAsObject enabled.

CROSS REFERENCE (Xref)

To use Xref, first select an object class from the drop down list, then highlight an object's name, and click Xref. The Xref feature searches the book to find pages that contain the object.

An object that is not referenced by any page is a candidate for deletion. However, use caution! It is possible a Program object might not be *contained* within any particular page, but still be *referenced* via a GOSUB command. XRef does not report such references. Before deleting a Program object, use the Find feature in the Book Editor to determine if such a reference exists.

To use a master object, refer to the Instance window.

Object Object

Applies to A-pex3 programming

Description The Object object is a generic way to refer to an object whose class is determined at run time. It is an experimental feature intended for use only by experienced programmers.

Details In certain situations, it is helpful to be able to refer to an object in A-pex3 programming without knowing the class to which it belongs, or its IDNumber. The following table describes the various Object "objects" currently available:

Object(6)	object that received the focus most recently
Object(7)	object most recently entered by the mouse cursor
Object(8)	object most recently exited by the mouse cursor
Object(9)	object most recently receiving a MouseDown event
Object(10)	object that generated an event most recently
Object(11)	object most recently dropped upon
Object(12)	object most recently dropped

Example Wthout Object, on a page that allows the user to drag and drop multiple objects, the DragDropEvent handler to move objects to their dropped location might resemble:

```
IF sysvar(113) = 84 & sysvar(114) = 1 THEN
  Textbox(1).Left = .Left+sysvar(9)-sysvar(159)
  Textbox(1).Top = .Top+sysvar(10)-sysvar(160))
ELSEIF sysvar(113) = 84 & sysvar(114) = 2 THEN
  Textbox(2).Left = .Left+sysvar(9)-sysvar(159)
  Textbox(2).Top = .Top+sysvar(10)-sysvar(160))
ELSEIF sysvar(113) = 86 & sysvar(114) = 1 THEN
  Picture(1).Left = .Left+sysvar(9)-sysvar(159)
  Picture(1).Top = .Top+sysvar(10)-sysvar(160))
ENDIF
```

However, with Object, this simplifies to:

```
Object(12).Left = Object(12).Left+sysvar(9)-sysvar(159)
Object(12).Top = Object(12).Top+sysvar(10)-sysvar(160)
```

Notes Since this is an experimental feature, please use it at your own risk.

Also see Obj() Function

Objects

Description Objects are the basic building blocks in Everest. Pages consist of one or more objects.

Objects
Animate
Button
Check
Combo
Erase
Flextext
Frame
Gauge
HScroll
HyperHlp
Include
Input
JLabel
Judge
Layout
Line
Listbox
Mask
Media
Menu
Object
OLE
Option
PicBin
Picture
Program
Shape
Special
SPicture
Textbox
Timer
VScroll
Wait

Also see Attributes, Commands, SaveAsObject

OLE Object

Description The OLE object provides Windows Object Linking and Embedding features.

Attributes
Action
BackColor
Bottom
Class
ClickEvent
Comment
Condition
Create
DblClickEvent
Destroy
DragMode
Execute
Focus
Format
GotFocusEvent
Height
IDNumber
Left
LostFocusEvent
MouseLeaveEvent
MouseOverEvent
Move
Name
Protocol
Right
SaveAsObject
ServerClass
ServerShow
ServerType
SetFocus
SourceDoc
SourceItem
Top
Update
Verb
Visible
Width
ZOrder

Details The OLE object provides a way to embed other applications within your project. The OLE features in Everest are intended for use only by experienced OLE programmers.

Also see Dde() Function, Shl() Function

OPEN Command

Applies to A-pex3 programming

Description Displays a page in another window, opening that window if necessary.

Syntax OPEN <PageName>

Details Use OPEN to run a page within another window. Note that unlike BRANCH and CALL, OPEN does not terminate or suspend execution of the current page.

Include the desired window number (1 to 8) within brackets immediately after the page name, for example:

```
OPEN help[2]
```

If the window is not already open, Everest creates it automatically.

OPEN A DIFFERENT BOOK

To use a page located in another Everest book (.ESL file), prefix the page name with the book name and a semicolon. Do not include the .ESL file name extension. For example:

```
OPEN lesson2;intro[2]
```

CLOSING AN OPENED WINDOW

To close a window opened via the OPEN command, employ the Destroy attribute. Do not use the RETURN command.

Notes If you want to place additional A-pex3 commands on the same line after the OPEN, separate them with a colon AND a space. For example:

```
OPEN thedoor: doors = doors + 1
```

You can open a maximum of 8 windows at a time, though you will likely reach the resource limits of Windows before reaching those of Everest. Use the Fre() function to monitor resources.

A Preview cannot open additional windows (such as via OPEN).

Also see BRANCH, CALL, Destroy, GOSUB, Include Object

Operators

Applies to A-pex3 programming

Description Operators are used to manipulate or compare two operands (number or string constants, variables or expressions) in programs.

Details Everest offers many operators, such as + for addition and - for subtraction. Most operators can be used with both numbers and strings. If both operands are numbers, a numeric operation is performed. If either operand is a string, a string operation is performed.

NUMERIC OPERATORS

+	addition
-	subtraction
*	multiplication
/	division
\	integer division
$	string creation
^	exponentiation
^/	modulo
^\	step function
^?	bit test (returns 0 or 1)
++	increment (numeric variable)
--	decrement (numeric variable)

Examples

Expression	Result
7 + 2	9
7 $+ 2	72 (prefix $ to force string operation)
7 - 2	5
7 * 2	14
7 / 2	3.5
7 \ 2	3
7 $ 2	77
7 ^ 2	49
7 ^/ 2	1
7 ^\ 2	4
7 ^? 2	0
varname++	add 1 to varname (faster than +1)
varname--	subtract 1 from varname (faster than -1)

STRING OPERATORS

+	concatenation
-	mid parse
*	search
/	right parse
\	left parse
$	string creation

^	replacement (see Notes below)
^*	count occurrences
^^	one middle character
^#	remove one or more characters

Examples

Expression	Result
"Eve" + "rest"	Everest
"Everest" - 2	verest
"Everest"*"e"	3
"Everest" / 2	st
"Everest" \ 2	Ev
"E" $ 2	EE
"Everest" ^* "e"	2
"Everest" ^^ 2	v
"Everest" ^# "Ee"	vrst
"Everest" - 2 \ 3	ver (also see Mid() function)

RELATIONAL OPERATORS

=	equals
>	greater than
<	less than
#	does not equal
>=	greater than or equal to
<=	less than or equal to
=E=	equivalent (ignore case and spaces)
#E#	not equivalent
=P=	pattern match
#P#	not pattern match
=S=	phonetic sound-alike
#S#	not phonetic sound-alike
=T=	inside region (see Reg() function)
#T#	not inside region
=W=	word search
#W#	not word search
&	AND (used between expressions in an IF command)
@	OR (used between expressions in an IF command)

PATTERN MATCHING

For the =P= and #P# pattern match operators, put the string that contains the pattern on the right side of the operator. Use the following symbols in the pattern to match as indicated:

?	any single character
*	0 or more characters
#	any single digit (0 to 9)
[list]	any single character in list
[!list]	any single character not in list

To specify a range in [list], use a hyphen; examples:

[a-z]	matches any character from a to z
[a-zA-Z]	matches a to z and A to Z

The following example tests if the single character in the variable named onechar is a vowel:

```
IF onechar =P= "[aeiouAEIOU]" THEN
```

WORD SEARCHING

For the =W= and #W# word search operators, put the word to search for on the right side of the operator. For example, the condition:

```
IF "washington" =W= "shin" THEN
```

is true because "shin" appears in "washington."

Example

Here's an example of operators in actual use (this code would appear in a Program object). This sample counts the number of files in the current subdirectory with the 8.3 file name extension .PCX:

```
filename = ext(41, "*.pcx")    $$ get first file
counter = 0                    $$ initialize
DO IF len(filename) > 0        $$ if found file
  counter++                    $$ increment counter
  filename = ext(42)           $$ get next file
LOOP
counter = "There are " + counter + " .PCX files."
dummyvar = mbx(counter, 0)
```

Notes

In an expression with multiple operators, the operations are performed from left to right (i.e. there is no precedence among operators). If desired, you can force precedence via (). So, for example, 3+4*5 returns 35, but 3+(4*5) returns 23.

The ++ and -- operators can only be used with variables that contain a numeric value, and may not be combined with other operators. For a usage sample, please see the variable named counter in the example above.

Parentheses (and/or functions) can be nested up to 8 levels deep.

You can force Everest to perform a string operation on two numeric operands by prefixing the operator with $. For example, 7$+2 returns 72.

The ^ string operator is used in a special way to replace portions of the string on the left side of the = sign. For example, in the expression:

```
longtext = "Everest" ^ 3
```

Everest copies the string "Everest" into the variable named longtext, starting at the third character position in longtext. So, if before the operation longtext contained the string

"AnxxxxxxxStory"

after the operation it would contain the string

"AnEverestStory"

Also see Attributes, Commands, Functions, IF, Program Object, Reg() Function, Rpl() Function

Option Object

Description Use several Option objects to display a group of items from which the user can select only one.

Attributes
Alignment
Answers1
Answers2
AntIncorrect1
BorderColor
BorderStyle
Bottom
BoxAlignment
BoxSize
Caption
CaptionColor
ClickEvent
CMIData
Comment
Condition
Create
Destroy
DragMode
EdgeSize
EdgeStyle
Enabled
FillColor
Font3d
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
GotFocusEvent
Group
GroupChoice
Height
IDNumber
Ignore
Initially
JudgeVar
Judgment
Left
LightColor
LostFocusEvent
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
MultiLine

Name
Pic
PicChecked
PicGrayed
PicPressed
PicUnchecked
Preset
ResponseVar
Right
SaveAsObject
SetFocus
ShadowColor
State
TabOrder
TabStop
Top
Tries
Update
Value
Visible
WallPaper
Width
Zev
ZOrder

Details

Option objects are typically arranged in groups of two or more items. When the user selects one item in the Group, the others in the same group are de-selected.

Some people call Option objects "radio buttons."

You can control the members of the group via the Group attribute.

Also see

Check Object

Orientation Attribute

Applies to	Media object
Description	Determines whether buttons on the media control panel are displayed horizontally or vertically.
Settings	0 horizontally 1 vertically
Details	If you want the buttons to be visible, make sure to enable the ShowButtons attribute.
Also see	ShowButtons

Other1Activator, Other2Activator, Other3Activator, Other4Activator, Other5Activator,Other6Activator, Other7Activator, Other8Activator Attributes

Applies to Wait object

Description Specifies the numeric event code that triggers the corresponding OtherAction.

Settings -32000 to 32000, or a string surrounded by quotes

Double click Opens event code dialog box. Press the desired key to automatically generate the corresponding event code.

Details Everest watches the events that occur in your project, and checks if one matches an event code you specify. If a match is found, the event is removed from the queue, and Everest performs the corresponding OtherAction. If the corresponding OtherAction is empty, Everest continues processing the page at the object that follows the Wait.

If your Wait object needs to detect more events that there are OtherActivators, use the AllOtherAction attribute to trap the remaining events.

Examples: The following Other1Activator triggers the Other1Action when the user presses the letter a (event code 65):

65

If desired, you can specify more than one event code per OtherActivator; to do so, separate with commas. The following example detects both a and Shift+A:

65, 1065

Also see OtherActions

Other1Action, Other2Action, Other3Action, Other4Action, Other5Action, Other6Action, Other7Action, Other8Action Attributes

Applies to	Wait object
Description	Specifies the action to perform when the corresponding OtherActivator event code is triggered.
Double click	Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.
Details	Enter any single line of A-pex3 programming code. When a Wait object sees that an event code matches an OtherActivator event, it traps that event code, and performs the corresponding OtherAction. If the corresponding OtherAction is empty, Everest continues processing the page at the object that follows the Wait. Most authors employ the OtherActivators and OtherActions on pages that have many possible events. For example, you might create a menu page for your project that branches when the user clicks on a button. If your page has many buttons, assign each button's ClickEvent a unique event code. Put the same event codes in the OtherActivators (one per OtherActivator), and use the OtherActions to execute individual BRANCH commands to the desired pages.
Notes	For delivery via the Inter/intranets, at run time, Everest automatically pre-downloads any granular pages named in BRANCH commands in the OtherActions.
Also see	OtherActivators

OutlineStyle Attribute

Applies to	Line, Shape objects	
Description	Sets the appearance of the edge of a shape or line.	
Settings	0	transparent
	1	solid
	2	dash
	3	dot
	4	dash-dot
	5	dash-dot-dot
	6	inside solid
Notes	For a non-solid border, you must set BorderWidth to 1.	
Also see	BorderWidth, DrawMode, FillStyle	

OUTLOOP Command

Applies to A-pex3 programming

Description Causes Everest to exit from inside a DO...LOOP.

Syntax OUTLOOP

Details Use OUTLOOP to immediately exit from inside a DO...LOOP, and continue processing after the LOOP command. Typically, OUTLOOP is used with an IF command.

Do not exit from a DO...LOOP via the JUMP command; use OUTLOOP instead.

Example The following example multiplies the elements of the array named values, and exits from the loop if the element counter exceeds the number of elements in the array:

```
counter = 0: mult = 1
DO
  counter++
  IF counter > arr("values") THEN OUTLOOP
  mult = mult * values(counter)
LOOP
```

Also see DO, RELOOP

Page Copier Window

The Page Copier Window is accessible via the main Author window's Utilities pull-down menu. The Page Copier helps you copy one or more pages from one book to another.

FROM (SOURCE)

In the source section, choose the book from which you are copying. Also, highlight the page(s) you wish to copy. Tip: to quickly highlight all pages of a book for copying, double click on the book's file name.

TO (DESTINATION)

Similarly, in the destination section, choose the book into which to copy the pages. To create a new book, type its name in the combo box. The file name extension must be .ESL. Any existing pages by the same name will be replaced.

Copy Objects Too - Enabling this tells Everest that if the page refers to any master objects (i.e. those with SaveAsObject enabled) to also copy the master object.

Copy External/Embedded Files Too - Enabling this tells Everest to scan the pages for any file references, and copy those files too.

Lock Pages - Enabling this tells Everest to lock the pages in the destination book to prevent their further editing. Use this to prevent anyone from modifying your pages. Careful! Maintain your own unlocked copy: there is no utility that unlocks locked pages!

OTHER NOTES

To copy a page within a given book (or even to another book in the same location), use the Book Editor.

Page Selection Window

The Page Selection Window appears when you use the File...Open or File...Save as features, as well as Run...Start at, and a few other features. The Page Selection Window helps you choose a page to employ.

DRIVES AND DIRECTORIES

Use the combo boxes and lists to select the directory in which you want to work. To open a directory (i.e. see the books within it), double click on it.

BOOKS

To open a book, double click on the book name (.ESL file) in the Books column. You can create a new book by clicking on the New Book button.

PAGES

To select a particular page, double click on it in the Pages column, or type the desired page name. Double clicking is the same as single clicking followed by clicking on the OK button.

FIND BUTTONS

The Find Text button searches for the text you specify within all pages of all books currently displayed in the Books column. It is handy if you recall a key word or phrase, but do not remember which book/page contained it.

The Find Page button searches for a particular page by name. It scans all the books currently displayed in the Books column.

NEW BOOK BUTTON

Use the New Book button to create a new book (a new .ESL file). You will be prompted to enter a name for the new book; enter up to 8 alphanumeric characters. The book will be created in the location (i.e. drive and subdirectory) highlighted in the Page Selection window when you clicked New Book. This location is shown within the window that prompts you to enter the new book name. If you would like to also create a new subdirectory to hold the new book, simply prefix the book name with the subdirectory name. For example, if the window shows the current location to be C:\EVEREST, if you enter `project1\book1`, Everest will create C:\EVEREST\PROJECT1\BOOK1.ESL.

DIRECT BUTTON

When you start a test run of a project, the Direct button lets you type the full location, book and page directly. This can be handy when entering an Inter/intranet URL. For example, you might enter something that resembles `http://www.insystem.com/evdemo/@start;@start`. Be sure to type your entry exactly as desired because Everest does not check it for illegal characters, improper case, etc.

PEEK BUTTON

To use the Peek feature, first click once on a page in the Pages column (to highlight it), then click on the Peek button. This quickly displays the page so you can determine if it is the one you want.

PAINT Command

Applies to A-pex3 Xgraphics programming

Description Fills a region bounded by a unbroken edge of a particular color.

Syntax PAINT (X, Y, [FillColor], [EdgeColor], [Style])

Details X and Y are the horizontal and vertical location, respectively, at which to begin painting. Express in pixels.

FillColor is the color to use for the paint. Express via a number from 0 to 15 to choose a palette color. If you omit the parameter, Everest employs the FillColor you selected via the COLOR command.

EdgeColor is the color of the boundary that tells Everest where to stop painting. Express via a number from 0 to 15 to choose a palette color. If you omit the parameter, Everest employs the foreground color you selected via the COLOR command.

Style is the paint style (solid, lines, etc.). Use a number from 0 to 7 (see the STYLE command for a description). If you omit the parameter, Everest employs the FillStyle you selected via the last STYLE command.

Example The following example draws a light blue triangle and fills it with solid dark blue:

```
STYLE (1, 4)$$ set line width
LINE (100, 200, 300, 200, 9)
LINE (200, 50,,, 9)
LINE (100, 200,,, 9)
PAINT (200, 100, 1, 9, 0)
```

Notes For proper operation, include a space between PAINT and (.

If your window unexpectedly becomes filled with paint, it is because either 1) you specified an incorrect EdgeColor, or 2) the edge has a break where the paint escaped.

In most cases, avoid using FillStyle 1 (Transparent). Transparent paint is invisible!

Due to a bug in Windows, PAINT has been known to fail on certain display adapters.

Also see GFILL, POLY, STYLE

PassChar Attribute

Applies to	Input object
Description	Sets the character to display instead of the text a user enters; useful for hiding a password.
Settings	Any single character; leave empty for normal operation of the Input object.
Details	The password feature works only when MultiLine is No, WordWrap is No, and Alignment is No.
Also see	Alignment, InputTemplate, MultiLine, WordWrap

Pause Attribute

Applies to	Wait object	
Description	Specifies the amount of time to wait for the next user action.	
Settings	Y	wait for user action
	N	do not wait, perform NextAction immediately
	>0 to 86400	pause for the number of seconds specified
Details	You can determine what event terminated the Pause via the EventVar.	
	When you enter N for Pause, EventVar is set to 0. When you enter a number for Pause, if the time expires, EventVar is set to the special code 32001. In both cases, the NextAction of the Wait object is performed.	
Notes	Use care when setting Pause to values other than Y (be sure a valid BRANCH eventually occurs, otherwise an inifinite loop can result).	
	Due to computer speed differences, the amount of time paused may differ from real time by up to approximately 10% or 10 seconds, whichever is less.	
Also see	EventVar, PAUSE	

PAUSE Command

Applies to	A-pex3 programming
Description	Halts execution of the project for a specified amount of time.
Syntax	PAUSE (Numeric)
Details	The Numeric value specifies the number of seconds to pause. Use a negative number for Numeric if you want the user to be able to cancel the pause via an event (i.e. by pressing a key, clicking a mouse button, etc.). The PAUSE command halts all processing, and can prevent a page or graphics command from plotting to completion. If this creates a problem in your project, try one of the following solutions: 1) reference the ext(101) function immediately prior to the PAUSE command; 2) use the Pause attribute of the Wait object instead; or, 3) use a Timer object.
Notes	For proper operation, include a space between PAUSE and (.
Also see	Pause

Period Attribute

Applies to	Timer object
Description	Specifies the amount of time between TimeEvents.
Settings	0 to 86400 (seconds)
Example	To generate a TimeEvent two times per second, enter 0.5 as the Period.

Details	The Timer keeps running and generating TimeEvents until you either erase the object from the window (via an Erase object or ERASE command), or disable it. To disable a Timer object with IDNumber 1, use either of the following A-pex3 commands:
	
	`Timer(1).Period = 0`
	
	`Timer(1).Enabled = 0`

Notes	Due to computer speed differences, the amount of time between TimeEvents may differ from real time by up to approximately 10% or 10 seconds, whichever is less.
Also see	TimeEvent

Pic Attribute

Applies to	Button, Check, Frame, Gauge, Option objects
Description	Specifies the image to display inside the object as the background.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).

Settings

a file name	display image file
0	clear image
>0	display PicBin image (cel number)

Details

The Pic attribute can load an image from either of two sources: 1) directly from a .BMP, .ICO or .WMF graphics file on disk, or 2) from a cel in the Picbin object.

To load an image from disk, enter the name of the file (you can double click on the attribute name, then click File Load to open the file load dialog). This approach is good when you have only a few such images to display in your project.

If you have many such images, consider pre-loading a group of them into a PicBin Object you place earlier in the page. Then, to copy an image from the PicBin into the current object, set Pic equal to the number of the cel that contains the image you want (you can double click on the attribute name to display the PicBin icons). The cels are numbered consecutively, left to right, then top to bottom, starting with 1 in the upper-left corner. This approach is best when you have many such images to display in your project.

To remove an image, set Pic to 0. To avoid loading or changing an image, leave Pic empty.

When it fills the entire object, the Pic image takes the place of the FillColor (background color) of the object.

Also see PicBin Object, Wallpaper

PicBin Object

Description — The PicBin object acts as a container for a hidden .BMP picture. Portions of the picture can be copied and displayed on other objects.

Attributes

BMPFile
Columns
Comment
Condition
Name
Rows
SaveAsObject

Details — Most authors use the PicBin object to hold a carefully prepared grid of icons and clip art. Each cel of the grid holds one image. The images can be displayed on Buttons and other objects by specifying a cel number for the Picxxx attribute of those objects.

You can use a graphics editor such as Windows Paintbrush to cut and paste all the (usually small) images you want onto a single .BMP, then load this .BMP into the PicBin via the BMPFile attribute.

You define the size of each PicBin cel (and therefore, the size of the image it contains) indirectly via the (number of) Columns and Rows attributes. Everest can automatically determine the size of the whole .BMP picture, so when you tell it the number of Columns and Rows, it can easily calculate the size of each cel. For example, if Everest determines that your .BMP is 320 pixels wide, and you set Columns to 10, it makes each cel image 32 pixels wide.

The sample icon files (such as FLAGS.BMP) provided with Everest are 32 x 32 pixels, arranged into 10 columns and 7 rows.

Each window can hold one PicBin picture at a time. The picture remains loaded from page to page until erased or replaced by another PicBin object. All cels in a PicBin are of the same size. The cels are numbered consecutively, left to right, then top to bottom, starting with 1 in the upper-left corner.

If desired, you can put multiple PicBin objects in one page. Recall that Everest processes the page from top to bottom. Each PicBin encountered determines the images displayed on subsequent objects in the page, until the next PicBin is encountered. Note that while editing a page that uses multiple PicBin objects, the VisualPage editor may be unable to determine the correct PicBin from which to obtain an image (i.e. objects might display an image from a different PicBin while editing).

Also see — Pic, PicChecked, PicDown, PicGrayed, PicPressed, PicUnchecked, PicUp

PicChecked Attribute

Applies to	Check, Option objects
Description	Specifies the image to display inside the box when the object is in a "selected" state.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).
Details	If you specify an image for PicChecked, you must as specify one for PicGrayed, PicPressed and PicUnchecked. Refer to the Pic attribute for settings and additional details.
Notes	.WMF files are not supported.
Also see	BoxSize, PicBin Object, PicGrayed, PicPressed, PicUnchecked, Wallpaper

PicDown Attribute

Applies to	Button object
Description	Specifies the image to display inside a Button object when the button is depressed.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).
Details	If you specify an image for PicDown, you must also specify one for PicPressed and PicUp, and not one for Pic. Refer to the Pic attribute for settings and additional details.
Also see	HoldDown, PicBin Object, PicPressed, PicUp, Wallpaper

PicGrayed Attribute

Applies to	Check, Option objects
Description	Specifies the image to display inside the box when the object is in a disabled state.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).
Details	If you specify an image for PicGrayed, you must also specify one for PicChecked, PicPressed and PicUnchecked. Refer to the Pic attribute for settings and additional details.
Notes	.WMF files are not supported.
Also see	BoxSize, PicBin Object, PicChecked, PicPressed, PicUnchecked, Wallpaper

PicPressed Attribute

Applies to	Button, Check, Option objects
Description	Specifies the image to display inside the object while the user is pressing it.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).
Details	Refer to the Pic attribute for settings and additional details.
Notes	.WMF files are not supported by the Check or Option objects.
Also see	PicBin Object, Wallpaper

Picture Object

Description — Use the Picture object to load and display images stored on disk in .BMP, .DIB, .ICO, .RLE and .WMF formats.

Attributes

AnimPath
AutoRedraw
AutoSize
BackColor
Bottom
ClickEvent
Comment
Condition
CopyBgnd
CopyPic
Create
DblClickEvent
Destroy
DragMode
Enabled
hDC
Height
hWnd
IDNumber
Initially
Left
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
Name
PictureFile
Right
SaveAsObject
SetFocus
SpecialEffect
Top
TpColor
Update
Visible
Width
Zev
ZOrder

Details — By default, the Picture object does not scale bitmapped images to fit the area. However, vector graphics (stored in .WMF files) *are* scaled to fit the area.

A unique feature of the Picture object is its ability to display an image with a special effect. See the SpecialEffect attribute for caveats. Images displayed with a SpecialEffect are scaled to match the area of the Picture object.

Another special feature of the Picture object lets you draw Xgraphics on top of the image. To do so, in an A-pex3 program, set Sysvar(108) to the IDNumber of the Picture object on which to draw.

Also see SPicture object

PictureFile Attribute

Applies to	Picture object	
Description	Specifies the image to display in the Picture object.	
Double click	Opens file dialog box. Double click on the file you want.	
Settings	FileName	displays the specified picture file from disk
	|FileName	displays the specified picture file from the book
	>0	displays PicBin image (cel number)
	<0	copies image from the Picture object with the IDNumber specified

Details The Picture object can display .BMP, .GIF, .ICO, .JPG, .PCX, .RLE, .TGA, .TIF, and .WMF format files. At the current time, animated .GIFs are not supported.

For help with file locations, refer to Appendix F.

Also see SpecialEffect

PicUnchecked Attribute

Applies to	Check, Option objects
Description	Specifies the image to display inside the box when the object is not in a "selected" state.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).
Details	If you specify an image for PicUnchecked, you must also specify one for PicChecked, PicGrayed, and PicPressed. Refer to the Pic attribute for settings and additional details.
Notes	.WMF files are not supported.
Also see	BoxSize, PicBin Object, PicChecked, PicGrayed, PicPressed, Wallpaper

PicUp Attribute

Applies to	Button object
Description	Specifies the image to display inside a Button object when the button is not depressed.
Double click	Opens icon dialog box (lets you visually choose an icon from the current PicBin), or opens the file dialog box (from which you can load a picture file).
Details	If you specify a PicUp image, you must also specify one for PicDown and PicPressed, and not one for Pic. Refer to the Pic attribute for settings and additional details.
Also see	PicBin Object, PicDown, PicPressed, Wallpaper

Pik() Function

Applies to A-pex3 programming

Description Returns one of a list of items stored within a delimited string, or the location of the item within the list. Can also find one or more items in the list.

Syntax pik(Which, List [, Find])

Details Express the list of items via the List parameter. Separate the items with an otherwise unique character (many authors employ a semicolon), and start the list with that character.

When the Find parameter is omitted, the action of the Pik() function depends on the value you specify via the Which parameter:

Which	Action
> 0	Returns the item number expressed by Which. The first item is number 1. If the value of Which is greater than the number of items in List, a null string is returned.
< 0	Returns the starting character position of the item expressed by Which. The first item is -1. If the item does not exist, 0 is returned.
0	Returns the number of items in the List.

When the Find parameter is specified, Pik() searches the List in a manner determined by Which:

Which	Action
1	Returns the item number where Find is found within List. Returns 0 if Find is not found. The search is case sensitive.
-1	Returns the starting character position of Find within List. Returns 0 if Find is not found. The search is case sensitive. (Faster than the search performed when Which = 1.)
0	Reserved for future use.

If the first character of the Find parameter is the list's separator character, Pik() counts the items found in the List. Always set Which to 1:

Which	Action
1	Returns the count of the number of items in Find that are found in List. The search is case sensitive.
other	Reserved for future use.

Examples

The following A-pex3 example Sets the variable named cloud to "Cirrus" because that is item number 2 in the list:

```
cloud = pik(2, ";Cumulus;Cirrus;Stratus")
```

The following example sets count to 3 because there are three items in the list:

```
count = pik(0, ";Cumulus;Cirrus;Stratus")
```

The following example sets cloudtype to 2 because Cirrus is found to be the second item in the list:

```
cloudtype = pik(1, ";Cumulus;Cirrus;Stratus", "Cirrus")
```

The following example sets charloc to 10 because Cirrus is found starting at the 10th character in the list:

```
charloc = pik(-1, ";Cumulus;Cirrus;Stratus", "Cirrus")
```

The following example removes the second item from the list:

```
clouds = ";Cumulus;Cirrus;Stratus"
ptr2 = pik(-2, clouds)
ptr3 = pik(-3, clouds)
IF ptr2 > 0 & ptr3 = 0 THEN   $$ second is last item
  clouds = clouds \ (ptr2-1)
ELSEIF ptr2 > 0 THEN
  clouds = (clouds \ (ptr2-1)) + (clouds - ptr3)
ENDIF
```

The following example sets x to 2 because two items in Find are found in the list:

```
x=pik(1,";Cumulus;Cirrus;Stratus",";Cirrus;Cumulus;Other")
```

Also see

ItemList, Listbox object, Rpl() Function

Play Attribute

Applies to Animate object

Description Determines whether the animation is active or stopped.

Settings

-1	play animation
0	stop animation
1	play animation at run time only

Example To allow the user to start and stop an animation by clicking on it, put the following in the Animate object's ClickEvent (all on one line):

```
p = Animate(1).Position: .Play = .Play + 1: .Position = p
```

Also see AnimFile, Position

Ply() Function

Applies to A-pex3 programming

Description Plays musical notes through the computer's speaker (no multimedia hardware, such as a sound board, is needed).

Syntax ply(Music)

Music is a character string of musical notes.

Details Specify Music by employing the following characters:

cdefgab	plays the indicated note in the current octave; optionally append a # or + for a sharp, or a - for a flat
o	sets the octave; suffix the letter o with a number from 0 to 6; octave 3 starts with middle c
L	sets the length of the following notes; suffix the letter L with a number from 1 (whole note) to 64 (64th note)
p	pause (rest/silence); suffix the letter with a number from 1 to 64
.	use after a note to play it as a dotted not (50% longer duration)
T	set the tempo (number of quarter notes per minute); suffix the letter with a number between 32 and 255; the default is 120

The function returns 0 is the Music string is ok; another value indicates an error.

To determine the number of notes waiting to be played, refer to ext(6). To stop music while it is playing, use ply("").

Example The following example plays the musical scale:

```
error = ply("o4 cdefgab o5 c")
IF error # 0 THEN
  dummyvar = mbx("Error in music!", 64)
ENDIF
```

Notes The Ply() function does not require a sound board or other multimedia equipment because it plays sounds through the computer's built-in speaker.

Also see Ext(6) Function, Mbx() Function, Mci() Function

POINT Command

Applies to A-pex3 Xgraphics programming

Description Draws a dot.

Syntax POINT (X, Y [,Color])

Details Use POINT to color a single pixel in the window. The X and Y parameters express the location of the pixel. Include the Color parameter only if you want to use one of the 16 palette colors; otherwise, Everest uses the current foreground color (which you can set via the COLOR command).

Example The following A-pex3 example draws dots in random locations and colors in the currently active window (window number 0) until an event occurs (keypress, etc.):

```
wwidth = Window(0).Width
wheight = Window(0).Height
DO IF ext(5) = 0
  POINT (rnd(wwidth), rnd(wheight), rnd(16))
LOOP
```

Notes For proper operation, include a space between POINT and (.

POLY Command

Applies to A-pex3 Xgraphics programming

Description Draws an irregular closed polygon containing 3 to 100 sides.

Syntax POLY (FillOption, VerticesString)

Details Use POLY to draw a polygon (a shape consisting of 3 to 100 line segments).

FillOption controls the type of paint operation. Use one of the following numbers for FillOption:

0 do not paint inside polygon

1 paint enclosed areas of polygon via the "alternate side" process

2 same as 1, except use "winding" paint process

VerticesString is a list of X-Y coordinates of the endpoints of the polygon's line segments. It must be expressed as a string constant or variable. Use the format: X1,Y1;X2,Y2;...;XnYn.

Example This example draws a red triangle filled with a solid white color:

```
COLOR (-1, 255, 0, 0)          $$ set edge color
COLOR (-2, 255, 255, 255)            $$ set fill color
STYLE (4, 0)                         $$ set fill style
POLY (1, "100,50;150,100;50,100")
```

Notes For proper operation, include a space between POLY and (.

Also see PAINT

PopupMenu Attribute

Applies to Layout object

Description Displays a menu at the current mouse location. Write only. Available at run time only.

Settings 0 to 7 the number of the column that contains the desired menu

Details Create the menu via the Menu object.

When the user chooses an item from the PopupMenu, the event code you specify via the Menu object is generated, and the menu is made invisible.

Example Most authors do not want the PopupMenu to be visible at the top of the window. To make the menu invisible, set the value of Checked for the non-indented caption to 0. For example, to prepare a PopupMenu that displays two items, Next and Previous, in the Menu object you would enter:

```
IgnoreMe,,,0
  Next, 34
  Previous, 33
```

In the example above, 34 and 33 are the event codes you wish to generate when the user chooses the menu item. Use a Wait object to detect and respond to the events.

At run time, to display the menu in this example, set PopupMenu to 0. For example, in the ClickEvent for the Layout object, you could enter:

```
Window(0).PopupMenu = 0
```

Also see Menu Object

PopupPointer Attribute

Applies to Flextext object

Description Controls the appearance of the mouse cursor while the cursor is positioned over a popup word. Accessible only at run time via A-pex3 programming.

Settings

0	default
1	arrow
2	cross-hairs
3	I-beam
4	icon
5	N, S, E, W arrows
6	NE, SW arrows
7	N, S arrows
8	NW, SE arrows
9	W, E arrows
10	up arrow
11	hourglass
12	"not allowed" symbol
13	hand
14	arm
15	target
16	wand
17	boy
18	girl
19	pencil
20	lightning
21	key
22	telephone
23	question mark

Also see JumpPointer, MousePointer, NormalPointer

Position Attribute

Applies to Animate, Input, Media objects

Description Returns or sets the current frame of an Animate object, returns or sets the location of the cursor in an Input object, or returns the current Media object location.

Details For a Media object, Position is read-only, describes the current location between StartAt and EndAt, and is expressed in TimeFormat units.

Example The following A-pex3 programming example plays an animation file (previously loaded into the Animate object) backwards:

```
frame = 50         $$ arbitrary number
DO
  Animate(1).Position = frame
  frame--
LOOP IF frame > 0
```

Preset Attribute

Applies to	Button, Check, Combo, HScroll, Input, Mask, Option, VScroll objects
Description	Controls the initial value of the object at run time.
Details	Use Preset when you want an interactive object to start with a certain setting (i.e. before the user has had an opportunity to change it) at run time. The Input and Combo objects expect the Preset to contain text. The other objects expect a number. Refer to the ResponseVar attribute to learn the range of possibilities for each object. To employ a variable or expression as a Preset, surround it with { }. Leave Preset empty if you do not want the object to start with a particular setting.
Notes	At run time, Everest reads the Preset attribute once, then clears it to avoid reuse should you JUMP to the object again.
Also see	ResponseVar

PRINT Command

Applies to A-pex3 Xgraphics programming

Description Displays text in the window.

Syntax PRINT ([X], [Y], [Color], Text [, ShadowColor] [, NextLine])

Details Use PRINT when you want to display simple text to the user. Everest considers PRINTed text to be a vector graphic, which means it appears in the layer that is behind objects. Be sure to consider DrawText as an alternative.

X is the horizontal pixel location at which to display the text (0 is the left edge of the window). If you omit X, the text is displayed starting where the last Xgraphic was drawn. Or, set X to the keyword "center" (including quotes) to make Everest automatically center the text horizontally.

Y is the vertical pixel location at which to display text (0 is the top edge of the window). If you omit Y, the text is displayed starting where the last Xgraphic was drawn. Or, set Y to the keyword "center" (including quotes) to make Everest automatically center the text vertically.

Color ranges from 0 to 15; it chooses one of the colors in Everest's 16-color palette. You can edit the colors in the palette via the COLOR command. If you omit Color, the text is printed using the window's current foreground color (which can be set via the COLOR command).

Unlike Color, ShadowColor is a specific color value, just like BackColor. Include ShadowColor only if you want a drop shadow for the text. For an alternate way of plotting text with a drop shadow, refer to DrawShadow.

By default, PRINT leaves the cursor at the end of the printed text. If you want the cursor to wrap to the next line, include a value of 1 as the NextLine parameter.

The text is displayed in the current font and font style. Use the FONT and STYLE commands to change the settings of subsequently printed text.

Examples

```
PRINT (100, 50, 14, "Hello")

PRINT (,,, "Hello")

PRINT ("center", "center", 14, "One moment, please...")

PRINT (1, 1, 14, dat(0))

PRINT (1, 1, 14, "Today is {dat(0)}.")

PRINT (1, 1, 14, "The shadow knows.", rgb(64, 64, 64))

PRINT (1, 1, 14, "Another shadow.", sysvar(36))

PRINT (1, 1, 14, "Go to next line", , 1)
```

Notes For proper operation, you must include a space between PRINT and (.

If you wish to determine the display length of text before actually PRINTing it, employ the Ext(109) function.

Also see DrawText, LPRINT, Textbox Object

Print Book Window

The Print Book window is accessible via the Author window's File pull-down menu. Use the Print Book window to generate a hardcopy (paper copy) of one or more pages of a book.

Pages to Print - Highlight the names of the pages to print. To highlight scattered pages, hold down the Ctrl key while clicking with the mouse. To quickly highlight all pages, double click with the mouse.

Print Page Image - Enable this feature to print an image of the page (i.e. what appears in the VisualPage editor when you open the page for editing). The image width and height determine the how much of the window to print. Due to Windows limitations, this image must be printed on a page (sheet of paper) of its own. Note: Due to a bug in Windows, do not enable this feature if you are running Windows in more than 256 colors.

Also Print Object Attributes - Enable this to print the attributes of the objects that make up each page. Enable **Print Attributes Even if Empty** if you want to print even those attributes in which you have not entered anything. Enable **Print Value Descriptions** to also print the text descriptions that accompany the settings of certain (usually numeric) attributes. Enable **Print Text Only** if you only want to print the text attributes; this can be handy for spell checking with an external program; when this is enabled, the contents of Flextext objects are printed without their special, embedded formatting characters.

Form Feed After Each Book Page - Enable this to start printing each page of your book on a new sheet of paper.

Print Button - Click this when ready to begin.

Setup Button - Click this to open the Microsoft Windows Printer Setup window (where you can select and configure a printer).

Close Button - Click this when done.

Program Editor

The Program Editor window is accessible by double clicking on a Program object icon in the Book Editor. In the Program Editor window you enter A-pex3 programming commands. While the bulk of most projects is created visually via dragging and dropping icons, certain specialized applications benefit by programming. For example, to create a simulation, you would want to use some programming.

Everest's programming language is called A-pex3. The syntax is somewhat similar to that of Microsoft's Visual Basic.

CREATING A PROGRAM

To create a program in your page, first drag a Program object icon in from the Toolset and drop it on either the VisualPage Editor or Book Editor. Then double click on the Program icon in the Book Editor. This opens the A-pex3 Program Editor window.

PROGRAMMING EXAMPLES

In the Program Editor, you might enter something that resembles:

```
score = lwr(100 * (correct / answered))
IF score >= 70 THEN
  passed = "Yes"
ELSE
  passed = "No"
ENDIF
GOSUB showscore
```

Hundreds of programming examples can be found in the technical reference/on-line help.

CHECK NOW

Use the Check Now feature to ask Everest to scan your Program for syntax errors. Syntax errors are those that are typically caused by typographical errors. Everest makes two passes: one with the interpreter and one with the compiler to make sure both deem your Program acceptable.

If the Syntax Checking feature is enabled, Everest automatically checks your Program when the focus leaves the Program Editor window.

TOGGLE

The Toggle feature is a very handy one for debugging. Toggle lets you easily convert lines of programming into comments (so Everest will ignore them at run time). Comments are prefixed with $$. Use Toggle again to convert the comments back into programming. To use the feature, first highlight the desired lines of programming, then choose Toggle.

EVALUATOR

Want to know the value a variable had at the end of the last test run? Highlight it and press Shift+F9. The Evaluator feature instantly computes the value of the highlighted expression. If you have not highlighted anything, it will attempt to evaluate the item at the current cursor location. This is a very

quick way to view the value of variables from a prior test run.

EXECUTING THE PROGRAMS

At run time, Everest executes the objects that make up a page, including Program objects, in the order in which they appear in the page.

It is also possible to execute Programs as subroutines via the GOSUB command. Many authors gather all commonly used Programs into a separate page. As long as SaveAsObject is enabled for a Program object, it can be accessed by any page in the book via GOSUB.

Program Object

Description — A Program object contains programming instructions written using the A-pex3 language syntax.

Attributes
Comment
Condition
Name
Refresh
SaveAsObject
Text

Details — Via the Program object, you can include specialized calculations, branching and graphics with your page. After adding a Program object to your page, you can open the A-pex3 program editor window by double clicking on the Program object icon in the Book Editor.

Items described in this technical reference as Commands can be entered in a Program object; for example:

```
PRINT (1, 1, 15, "Hello world")
```

Items described in this technical reference as Attributes can be manipulated in a Program object; for example:

```
Shape(1).Shape = 3      $$ a circle
```

The $$ signs are used to prefix comments. Everest ignores anything that follows the $$ on a line.

A-PEX3 COMPILER

To boost execution speed, Everest's just-in-time compiler automatically compiles some of your Program objects. Compilation is performed at run time if ANY one or more of the following conditions are true for the Program object:

1) it is part of a preloaded page (see NextAction)
2) it contains a DO...LOOP
3) it is part of a page being test run via AUTHOR's Preview feature
4) it starts with the compiler directive `$$compile`

and provided the Program object does not start with the compiler directive

```
$$nocompile
```

The `$$nocompile` directive should rarely be needed. Authors sometimes use it when they suspect a problem with the compiler, or want to compare execution speeds of non-compiled and compiled Programs. The compiler runs Programs an average of 3 times faster.

Notes — If you want to be able to access a Program object from another page via the GOSUB

command, be sure to set SaveAsObject to Yes.

Program objects are limited to 32K characters in size. Since the number of characters stored locally with a page is also limited to 32K characters, several large Program objects in the same page can cause trouble (error 255). To avoid this problem, set the SaveAsObject attribute of large Program objects to Yes.

Also see GOSUB, SaveAsObject, Visual Basic

Project Packager Window

The Project Packager Window is accessible via the main Author window's Utilities pull-down menu. The Project Packager helps you prepare your projects for distribution to end users. It gathers together all the files an end user will need to run your project.

FROM (SOURCE)

In the source section, choose the location (i.e. disk path) that contains your project. Also, highlight the book(s) you wish to copy. All pages of the book will be copied.

Copy Graphics/External Files - Enable this feature to also gather together any external files (such as graphics) that your project uses. In most situations, you want to enable this feature. Note that the Packager is very aggressive when looking for possible files: any text in your project that resembles a typical 8.3-style file name will be considered for copying. If it cannot find the corresponding file for a name, it will display a message to this effect, and give you an opportunity to specify a different source file, or skip the file.

Copy Everest Run-time Files - Enable this feature to also gather together Everest's run time player files. These are the Everest files that perform the playback of your project. Note: the free version of Everest does not include an important run-time file: ERUN. To obtain ERUN, you must first license Everest.

Compress Together into .ZIP Format - Enable this feature to automatically compress the packaged files into .ZIP form. The files can be uncompressed with the PkUnZip shareware utility. Everest's run time files will be compressed into files named ERUN1.ZIP, ERUN2.ZIP, etc. Your project will be compressed into files named PROJ1.ZIP, PROJ2.ZIP, etc. If files by these names already exist, they will be overwritten. If you wish, after packaging, you can rename these files via DOS or the Windows File Manager.

Compress Each File into .ZIP Format - Enable this feature to automatically compress individual project files into .ZIP form. Use this feature to prepare your project to take advantage of Everest's auto-ZIP check feature. Note that the auto-ZIP check feature operates only when running your project; AUTHOR cannot directly edit projects stored in .ZIP files.

TO (DESTINATION)

In the destination section, choose the location in which to store the packaged project. Sometimes this is called "a staging area." Most authors first create a new subdirectory on their hard disk to act as a staging area. Then they use Everest's Packager to put the packaged project into this staging area. Finally, to make a copy for the end user, they simply use DOS or the Windows File Manager to copy the packaged files from the staging area onto diskettes or CD-ROMs.

Max Chunk Size in K Bytes - Use this feature to place a maximum limit on the total size of the files the Packager gathers together in one spot. Each "K" equals 1024 bytes, so for example, if you are packaging onto 1.44 mb floppy diskettes, you would enter `1440` into this field. If you are packaging into a staging area (and not using .ZIP format), the Packager will create nested subdirectories, each containing files up to the size limit you specify. If you have no size limit, enter 0 in this field. Special feature: if the K limit varies from distribution disk to disk (if, for example, 440K of the first distribution diskette will be occupied by an installation program), enter a negative value in the field (for example, `-1000`); doing so tells the Packager to prompt you to enter a new limit for the next disk (so you can increase it for a

completely empty diskette if you want).

Make Granular - Enable this feature if you want the Packager to create a separate book for each page. Typically, you enable this feature only when preparing your project for Inter/intranet delivery. The names of the books are determined by the first eight letters of each page name. If the page invokes external Program objects via GOSUB, the Packager automatically copies the referenced Program object into the book. However, if the referenced Program object itself invokes other Programs via GOSUB, these will not be copied, and will cause errors during run time. Avoid using such nested GOSUBs when you will be employing the Make Granular feature.

Password Protect Books - This feature is available when you make pages granular. When you enable it, Everest will ask you to enter a password to assign the granular books. Later, if you attempt to open the book for editing, Everest will request the password. This feature handy if you are distributing your projects via the Internet and do not want others to be able to modify them.

Lock to Prevent Further Editing - This feature is available when you make pages granular. If enabled, it prevents further editing of the pages *by anyone*. This is handy if you are distributing your projects via the Internet and do not want anyone to be able to modify them. NOTE: Once a page is locked, there is no facility for anyone, including you, to unlock it! Be sure you maintain a separate, unlocked version for your own use.

Generate Setup (User Install) Script - Enable this feature to have the Packager create a setup file that is compatible with InstallShield Corp.'s InstallShield compiler. You will need a copy of InstallShield to make use of the script the Packager generates.

Starting Book - Use this feature to designate the book in which the end user will start. The Packager puts the contents of this field into the ProjectFile= item of the EVEREST.INI.

User Comments File - Enter the location (i.e. disk path) and name of the file that will hold user comments. For example, `USERCMTS.ECM`. The file name extension must be .ECM. Omit the path if you want Everest to write the comments in the same location as the project. The Packager puts the contents of this field into the Comments= item of the EVEREST.INI.

Enable User Log on and Records - Enable this feature if you want to employ Everest's built-in user recordkeeping feature (with bookmarks) for your project.

User Records File - Enter the location (i.e. disk path) and name of the file that will hold user records. For example, `USERRECS.EUR`. The file name extension must be .EUR. Omit the path if you want Everest to write the records in the same location as the project. The Packager puts the contents of this field into the UserRecs= item of the EVEREST.INI. Each user records file can store bookmarks for up to 32,000 different users.

PromptChar Attribute

Applies to	Mask object
Description	Determines the character used in place of InputTemplate symbols to alert the user that a character is to be entered.
Settings	any single character
Also see	InputTemplate

Protocol Attribute

Applies to	OLE object	
Description	Determines the protocol used when a new object is created.	
Settings	StdFileEditing	creates an object that can be edited by the user, and that can send Execute strings to the server.
	StdExecute	creates an object that can send Execute strings to the server.
	Static	creates an object that can be edited until the server application is closed.
Details	Most applications support only StdFileEditing.	
Also see	Action, Execute	

Pth() Function

Applies to A-pex3 programming

Description Returns a selected portion of a file name, or computes Everest drive letter designators.

Syntax pth(Operation, FileName)

Details Pth() parses a file name and returns portions such as disk drive letter, directory, 8.3 file name extension, etc. You select an operation via the Operation parameter:

Operation	Returns
1	drive only (including colon)
2	path only (including backslashes)
3	drive + path
4	base file name
5	file name extension
6	base file name + extension
-1	computes drive letter designators ?, @, *, &, ^ and % ? = path to current book @ = DOS default path * = Star Path (defined in EVEREST.INI) & = path to Windows directory ^ = path to EVEREST.INI % = path to current Everest .EXE program ~ = address of Inter/intranet source

Examples The following A-pex3 programming examples show what Pth() returns for each Operation:

Expression	Returns (example)
`pth(1, "C:\dir\file.ext")`	`C:`
`pth(2, "C:\dir\file.ext")`	`\dir\`
`pth(3, "C:\dir\file.ext")`	`C:\dir\`
`pth(4, "C:\dir\file.ext")`	`file`
`pth(5, "C:\dir\file.ext")`	`ext`
`pth(6, "C:\dir\file.ext")`	`file.ext`
`pth(-1, "&:system.ini")`	`C:\WINDOWS\system.ini`

`pth(-1, "~:a.jpg")`	`http://www.xyz.com/a.jpg`

Also see Fyl() Function

QuitAction Attribute

Applies to Wait object

Description Specifies the action to perform when the QuitActivator event is triggered.

Double click First: sets QuitAction to BRANCH @end. Second: sets QuitAction to BRANCH @finish. Subsequent: Opens page name dialog box. Double click on the name of the page to which to branch, and Everest will automatically create the proper BRANCH command for you.

Details Enter any single line of A-pex3 programming code.

When a Wait object sees that an event code matches the QuitActivator event, it traps that event code, and performs the QuitAction.

Most authors employ the QuitActivator and QuitAction to trap a user's request to exit from the project.

Before performing a QuitAction, Everest displays EVEREST.MSG number -32 in a window to verify that the user wants to quit. To disable this feature, edit the EVEREST.MSG file and delete the text portion of message -32.

Example While you can perform any QuitAction you want, the most common ones are the following:

`BRANCH @finish`	saves the user's bookmark, and branches to a page named @finish
`BRANCH @end`	saves the user's bookmark, and terminates the application
`BRANCH @exit`	terminates the application without saving the user's bookmark

Also see QuitActivator

QuitActivator Attribute

Applies to	Wait object
Description	Specifies the numeric event code that triggers the QuitAction.
Settings	-32000 to 32000, or a string surrounded by quotes
Double click	Opens event code dialog box. Press the desired key to automatically generate the corresponding event code.
Details	Everest watches the events that occur in your project, and checks if one matches the event code you specify as the QuitActivator. If a match is found, the event is removed from the queue, and Everest performs the QuitAction. Most authors employ the QuitActivator to detect when a user wishes to exit the project.
Example	To make a Ctrl+Q keypress the event that invokes the QuitAction, set the QuitActivator to the event code for Ctrl+Q: 2081.
Also see	QuitAction, Wait Object

RBOX Command

Applies to A-pex3 Xgraphics programming

Description Draws a rectangle with rounded corners.

Syntax RBOX (X1, Y1, X2, Y2 [, Color] [, Roundness])

Details The RBOX command draws rectangle with rounded corners in the window. Specify the coordinates of two opposite corners in pixels via the X1, Y1, X2 and Y2 parameters. Roundness indicates how rounded the corners are; use a value between 0 (sharp corner) to 100 (ellipse). Include the Color parameter only if you want the box to be drawn using one of the 16 palette colors; otherwise Everest uses the current foreground color set via the COLOR command.

Example The following example draws a rounded box with a dashed outline in palette color 13:

```
STYLE (1, 1)      $$ pen width must be 1
STYLE (3, 1)      $$ select dashes
RBOX (50, 50, 100, 100, 13, 30)
```

Notes Due to a bug in Windows, RBOX does not work well when the box size is small.

For proper operation, include a space between RBOX and (.

The interior of an RBOX can be filled via PAINT.

Also see BOX, STYLE

Rec() Function

Applies to A-pex3 programming

Description Reads and/or writes information from/to the user records and CMI databases.

Syntax ecode = rec(Operation [, "Variable"])

Details Most projects employ Everest's default log on and log off methods. By default, Everest automatically maintains the user records and CMI databases; therefore, most authors never need the Rec() function. However, for specialized applications, such as a custom log-on page, the Rec() function can be handy, or even essential.

The Rec() function lets you access Everest's user records databases at run time. The two most common uses of Rec() are to save custom CMI data, and to create a custom user log on. These topics are discussed in more detail in the Design Guide.

If Rec() is successful, it returns -1, otherwise Rec() returns a numeric error code (see Appendix C for error code interpretation).

Operation	Result
-4	Write variable to CMIFILE.DAT. Specify the variable name surrounded by quotes. For example: `ecode=rec(-4, "saveme(3)")` The CMIFILE.DAT contents can be processed with the SUMCMI.EXE program.
2	Save user bookmark (variables, place in project, etc.) to disk. Handy if you want to forcibly update the user's bookmark prior to log off.
3	Save user's author defined variables to disk. Bookmark and Sysvars are not also updated.
6	(Re)load user's bookmark and variables. For use only from an @preempt page. Do not assign the result to a variable, instead use the function in an IF statement, such as: `IF rec(6)#-1 THEN BRANCH @start`
7	Log on user whose name is specified in the Sysvar(131) to Sysvar(134) variables. Returns 0 if user record does not exist, 1 if user cancelled password entry, 2 if user exhausted password tries, 3 if Sysvar(132) (last name) is empty, 9 if password in Sysvar(129) is wrong, -1 if log on attempt is successful, or -2 if another error occurred (error code returned in Sysvar(1).
8	Initialize new user record. Use only after checking that Rec(7) returns 0 (i.e. user record does not yet exist).

9	Same as 7, except automatically initialize new user record if one does not already exist.
10	Delete current user's record from database. No bookmark will be saved upon this user's log off.
12	Uploads the user records file to the FTP site designated by the various FTP items in the EVEREST.INI file. The file is uploaded with a name that consists of the first letter of the user's log on last name, plus seven random characters, plus the extension .EUR. This unusual format is employed to virtually elminate the possibility that two different users will upload a records file with the same name. The files can be viewed with the Everest INSTRUCT administrator program.
13	Uploads the CMI data file to the FTP site designated by the various FTP items in the EVEREST.INI file. The file is uploaded with a name that consists of the first letter of the user's log on last name, plus seven random characters, plus the extension .DAT. This unusual format is employed to virtually elminate the possibility that two different users will upload a records file with the same name.
14	Verifies that the current user's log on information exists in the access list (.the .EAL file). Returns 0 if the user is not in the access list, some other number otherwise.
107	Same as 7, except if the user's record already exists, automatically resume where last logged off (i.e. do not provide user with choice).
109	Same as 9, except if the user's record already exists, automatically resume where last logged off (i.e. do not provide user with choice).

Example

The following example uses the Rec() function to manually add information to the CMI database. Since CMIData saves this information automatically, rarely should you need to use programming such as that below. The basic technique involves putting the desired data into a variable, then referencing the Rec() function to write that variable into the CMI file. The following A-pex3 program writes the user's Button choice, as well as the answer judgment, into the CMI database:

```
$$ after a Wait, sysvar(12) has most recent event code
$$ this example assumes the following ClickEvent values
$$ -65 for Button(1)
$$ -66 for Button(2)
$$ -67 for Button(3)
$$ -68 for Button(4)
$$ so, first convert event code to ID# 1, 2, 3 or 4

clickid = abs(sysvar(12)) - 64

$$ next, get the Caption of this Button
$$ Caption will be A, B, C or D
```

```
choice = Button(clickid).Caption

$$ write it to disk via Rec()
ecode = rec(-4, "choice")

$$ = -1 means no error, proceed
IF ecode = -1 THEN

  $$ determine whether correct/incorrect
  $$ by checking if clicked Button
  $$ has correct answer of -1
  IF Button(clickid).Answers1 = "-1" THEN
    judgment = 1
  ELSE
    judgment = 0
  ENDIF

  $$ write that judgment to disk
  ecode = rec(-4, "judgment")
ENDIF

$$ if error, report it
IF ecode # -1 THEN
  ecode = "Error " + ecode
  dummyvar = mbx(ecode + " writing CMI data")
ENDIF
```

Notes

The Rec() function operates only at run time (i.e. it is ignored while you test run pages via the AUTHOR program).

Rec() employs the user records file named in the UserRecs item in the EVEREST.INI. The user records file must have a file name extension of .EUR.

As part of a user's bookmark, Everest may also need to save the visual content of the window(s) and other image related objects. It does so in files with .SPW, .EPW, and .SP0 through .SP9 file name extensions.

Also see

CMIData

REDIM Command

Applies to	A-pex3 programming
Description	Changes the number of elements in an array, or creates the array if it does not yet exist.
Syntax	REDIM Arrayname(NewElements)
Details	ArrayName is the name of the array to change. NewElements is the new number of elements in the array. Use REDIM to change the size of (number of elements in) an array. The values of existing elements in the array are not changed. If you want the values of the array elements to be discarded, use DELVAR before REDIM. Relatively speaking, REDIM is a slow command. Avoid using it where speed is important.
Example	`REDIM files(100)`
Also see	Arr() Function, DELVAR, DIM, Var() Function

Refresh Attribute

Applies to Program object

Description Controls whether a Program object is re-executed when a window is repainted.

Settings
Yes re-execute the Program object upon repaint
No do not re-execute the object upon repaint

Details It can be appropriate to enable the Refresh attribute when the Program object contains A-pex3 programming you want to automatically re-execute when Windows tells Everest that the window must be repainted. Xgraphics commands (such as LINE and CIRCLE) are examples of programming that typically needs to be re-executed.

Xgraphics commands often need to be re-executed after another, obscuring window is removed from the display. This is because Windows does not automatically replot the Xgraphics in the window, as it does objects such as Textboxes and Shapes. Instead, Windows sends a message to Everest that the graphics in the window need refreshing. Everest then passes this message along to your project (by re-executing Program objects for which Refresh is set to Yes).

Immediately before re-executing a Program object, Everest sets Sysvar(126) to the number of the window being repainted, then back to 0 when done. If necessary, you can refer to Sysvar(126) inside your Program object to isolate programming that should or should not be re-executed.

Windows has several bugs and quirks handling Refresh; therefore we recommend its use only by experienced authors. For an easier and more reliable (but more memory intensive) way to automatically refresh Xgraphics, refer to the AutoRedraw attribute of the Layout object.

Notes When Windows sends Everest a repaint message, Everest re-executes only those Program objects that: 1) are in the current page, and 2) have Refresh enabled.

Due to a Windows limitation, do not use the Ext(101), Ibx() or Mbx() functions within a Program object that has Refresh set to Yes.

When Refresh is enabled, also be sure Program object's Condition is set to -1.

Windows tells Everest when to repaint a window; unfortunately, Windows often repaints more often than is needed. To help avoid excess repaints, put Program objects (that have Refresh enabled) immediately before a Wait object.

To tell Everest to ignore one or more repaint messages from Windows, set Sysvar(153) to the desired number of repaint messages to ignore. When a repaint message is ignored, Everest decrements Sysvar(153).

Also see AutoRedraw, Update

Reg() Function

Applies to A-pex3 programming

Description Converts X-Y coordinate pairs into a string that can be used by the =T= operator for area comparisons.

Syntax reg(X1 [, Y1 [, X2, Y2]])

X1 and X2 are numbers that represent horizontal display locations, expressed in pixels, relative to the left edge of the window.

Y1 and Y2 and numbers that represent vertical display locations, expressed in pixels, relative to the top of the window.

Details The Reg() function coupled with the =T= operator lets you determine if a point is located with a rectangular area. This is handy when you want to know if a user has clicked the mouse within a region.

The Reg() function converts one or two pairs of X-Y coordinates into a region string. Region strings can be compared with the =T= operator.

If you include only the X1 parameter, Reg() assumes X1 is a string of two coordinate pairs in the form returned by the Move attribute, specifically: Left, Top, Width, Height. Reg() takes this string and converts it into the form: Left, Top, Right, Bottom (the form required by the =T= operator).

Examples The following example checks if the last mouse click was located within a small rectangular area with top-left corner located at window coordinate 50, 60, and bottom-right at 70, 90:

```
x1 = sysvar(9)          $$ last mouse click X
y1 = sysvar(10)   $$ last mouse click Y
IF reg(x1, y1) =T= reg(50, 60, 70, 90) THEN
  dummyvar = mbx("Inside!", 128)
ELSE
  dummyvar = mbx("Outside", 128)
ENDIF
```

In the example above, reg(50, 60, 70, 90) returns the string "50,60,70,90". You could have instead actually entered "50,60,70,90" in your program. The Reg() function is of most help when the X-Y coordinate values are stored in variables since such values are tedious to convert into a region string of the correct syntax.

The following example checks if the mouse pointer is currently positioned over the Button with IDNumber 1:

```
IF reg(mse(1), mse(2)) =T= reg(Button(1).Move) THEN
  dummyvar = mbx("Over the button!")
ENDIF
```

The following example determines the location of the Button with IDNumber 1 relative

to the Frame with IDNumber 1:

```
frameat = reg(Frame(1).Move)
IF reg(Button(1).Move) =T= frameat THEN
  dummyvar = mbx("Button is inside Frame.")
ELSEIF reg(Button(1).Left, .Top) =T= frameat THEN
  dummyvar = mbx("Button is on edge of Frame.")
ELSEIF reg(Button(1).Right, .Bottom) =T= frameat THEN
  dummyvar = mbx("Button is on edge of Frame.")
ELSE
  dummyvar = mbx("Button is outside Frame.")
ENDIF
```

Also see Move, Mse() Function

Relocate Attribute

Applies to Layout object

Description Controls whether the location and size of the window on the display are updated.

Settings
Yes relocate the window
No do not relocate the window

Details When a new page is shown in an existing window, this attribute controls whether the location and size of the window are updated to reflect the values in the Left, Height, Top, Width and WindowState attributes.

If you are allowing the user to freely move and size the window manually, you should set the Relocate attribute to No. This lets the window stay where the user has placed it as different pages are shown within.

The very first time a new window is opened, Everest ignores the Relocate attribute, and always places the window as specified in the Left, Height, Top, Width and WindowState attributes.

Also see AutoCenter, Left, Height, Top, Width, WindowState

RELOOP Command

Applies to	A-pex3 programming
Description	RELOOP is used inside a DO...LOOP to force processing back to the top of the loop.
Details	RELOOP branches back to the DO command. If the DO command includes a condition, that condition is reevaluated, and if false, the loop is exited.
Also see	DO, OUTLOOP

RemoveItem Attribute

Applies to Combo, Listbox objects

Description Deletes an item from the list. Write-only and available at run time only.

Settings
-1 delete all items from the list
>= 0 delete a specific item from the list

Details Set RemoveItem equal to the number of the Item to delete. The items are numbered starting with 0 at the top.

Example The following example deletes all items from the Combo object with IDNumber 1:

```
Combo(1).RemoveItem = -1
```

The following example deletes the highlighted items from the Listbox with IDNumber 1:

```
marked = Listbox(1).TaggedList
ptr = len(marked)
DO IF ptr > 0
  IF marked ^^ ptr = "X" THEN .RemoveItem = ptr - 1
  ptr--
LOOP
```

Also see AddItem, TaggedList

ResizeEvent Attribute

Applies to	Layout object
Description	Event code to generate, or programming to perform, when the window size (Height and/or Width) changes.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	When you enter a number or string constant for ResizeEvent, you are merely telling Everest what event to generate when the size of the window changes. To make use of that event (i.e. detect it and do something useful, like rearrange objects in the window), you must include a Wait object in your page.
Notes	As the corresponding action for a ResizeEvent, do not close the window.
Also see	CloseEvent, MoveEvent

ResponseVar Attribute

Applies to — Button, Check, Combo, HScroll, Input, Listbox, Mask, Option, VScroll objects

Description — Specifies the name of the variable in which Everest stores the user's response upon judgment (encountering a Judge Object) at run time. Do not surround the variable's name with { }.

Details — The information stored into the variable depends on the class of the object. Here is a table:

Button	-1 (pressed) 0 (not pressed)
Check	0 (unchecked) 1 (checked) 2 (dimmed)
Combo	text in the box on top
HScroll	numeric value between Min and Max
Input	text of the user's response
Listbox	list of selected items (TaggedList format)
Mask	the text in the object
Option	-1 (selected) 0 (not selected)
VScroll	numeric value between Min and Max

Many authors store the user's response in order to display it later, or to test it in an IF...THEN conditional expression.

For Input objects, the user's response is returned as is (i.e. spaces and the case of the letters are preserved). To make the response easier to compare in a conditional expression, remove the spaces and convert to lower case.

Example — The following example removes spaces and converts the contents of the useranswer variable to lower-case:

```
useranswer = lwr(useranswer ^# " ")
```

Notes — ResponseVar is filled only upon encountering a Judge object in the page. To determine the user's response without employing a ResponseVar, use A-pex3 code to examine the contents of the appropriate object attribute (for example, Input(1).Text).

Everest does not require that you specify a ResponseVar for proper operation of answer judging.

Also see Judge Object, JudgeVar, Preset

RETURN Command

Applies to A-pex3 programming

Description Causes execution to resume at the location of the most recent CALL command.

Syntax RETURN

or

RETURN [closewindow#]

Details A plain RETURN resumes at the most recent CALL command, leaving the current window open.

A RETURN followed by a number surrounded by [ ] does the same, but also closes the indicated window number (0 to 8). Use window number 0 to easily refer to the current window.

Example The following example returns from the most recent CALL and closes the current window:

```
RETURN [0]
```

Note An error occurs if you attempt to RETURN without having used a corresponding CALL.

Also see CALL

Rgb() Function

Applies to	A-pex3 programming
Description	Converts a color value specified via its red, green and blue components into a single number form that is useful for assigning color object attributes.
Syntax	rgb(Red, Green, Blue)
Details	Red, Green and Blue are numeric values in the range 0 to 255 that express the desired amount of that primary color. Or, to obtain the nearest solid color, use a negative value (-1 to -255) for Red, Green and/or Blue.
Example	The following example sets a Button object's FillColor to blue-green (cyan): `Button(1).FillColor = rgb(0, 192, 192)`
Also see	BackColor, COLOR

Right Attribute

Applies to	Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects
Description	Controls the location of the right edge of the object. Available at run time only.
Settings	A value larger than that in the Left attribute.
Details	Specify in units of pixels. The value of Right is the same as Left + Width.
Also see	Left, Move, Width

Rnd() Function

Applies to	A-pex3 programming
Description	Returns a random number greater than or equal to 0 and less than Numeric.
Syntax	rnd(Numeric)
Details	The accuracy (number of decimal places) of the number returned matches that you specify in Numeric.
Example	The following example stores a randomly chosen number greater than or equal to 0 and less than 11 in the variable named quiznum: `quiznum = rnd(11)`
Notes	When debugging, the series of random numbers will always be the same each time you run the AUTHOR program. In ERUN, the random numbers are different.
Also see	Sel() Function, Sfl() Function

Rows Attribute

Applies to	PicBin object
Description	Sets the number of columns into which to divide the image loaded into the PicBin.
Example	If the image in the PicBin is 320 pixels high, and the height of each icon in the image is 32 pixels, you would set the Rows attribute to 10 (320/32).
Also see	Columns, PicBin Object

Rpl() Function

Applies to A-pex3 programming

Description Searches for all instances of a particular character string within another string, replaces them with a third string, and returns the modified string. Can also remove all instances of a particular character string from a string.

Syntax rpl(String, OldString [, NewString])

String is the string that contains the characters to be replaced or removed.

OldString is the string to replace or remove. Express OldString as either the actual character string, or (for a single character) via its numeric ASCII code.

NewString is the string to use in place of OldString. Express NewString as either the actual character string, or (for a single character) via its numeric ASCII code. If you omit NewString, Everest does not replace OldString, instead it removes all instances of OldString from String.

Examples The following example removes all Tab characters (ASCII 9) from the text in the Input object with IDNumber 1, and stores the result in the variable named notabs:

```
notabs = rpl(Input(1).Text, 9)
```

The following example converts all backslash characters to slash characters in the variable named slash:

```
slash = rpl(slash, "\", "/")
```

The following example replaces "DOS" with "Windows" in the Textbox with IDNumber 1:

```
Textbox(1).Text = rpl(Textbox(1).Text, "DOS", "Windows")
```

Also see Operators, Pik() Function

Rtr() Function

Applies to	A-pex3 programming
Description	Returns the character string String with trailing spaces (ASCII 32) removed.
Syntax	rtr(String)
Example	The following example removes the spaces (if any) from the end of the string stored in variable mytext: `mytext = rtr(mytext)`
Also see	Ltr() Function, Rpl() Function

SaveAsObject Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, Menu, OLE, Option, PicBin, Picture, Program, Shape, SPicture, Textbox, VScroll, Wait objects

Description — Controls whether the object is stored in the library independently of the screen, thereby making it easy to access from other screens. Read-only at run time.

Settings

Yes	save independently of screen
No	save locally with screen only

Details — When an object is stored independently in the library, it can be accessed from more than one screen (via object instancing, or, for Programs, via a GOSUB command).

When stored locally (i.e. stored completely within the IconScript), the object can only be accessed by that one screen. The advantage is that objects stored locally occupy approximately 50% less disk space on average.

To employ an instance of an object previously saved into the library with SaveAsObject enabled, use the Instance of... feature in the Attributes pull-down menu. Alternatively, hold down the Ctrl key while dragging and dropping an object icon from the ToolSet; when you drop the icon, Everest will display a list of available objects to instance.

Notes — When you edit an object for which SaveAsObject is set to Yes, Everest changes the background color of the Attributes window. This is to remind you that other screens might use this object; changing the non-instance attributes of the object will influence how it appears/operates on these other screens.

When SaveAsObject is set to No, avoid assigning an object the same Name as another object. Doing so will confuse Everest (it won't know if you want to share attributes among the like-named objects), and will likely confuse you if you later change SaveAsObject to Yes.

Similarly, if the object was previously saved into the library with SaveAsObject set to Yes, if you change SaveAsObject back to No, also change the Name of the object to avoid confusion with (and possible deletion of) the independent copy of the object in the library.

Everest can store approximately 30000 characters in the IconScript of each screen. Complex screens with a large amount of text could exceed this limit if SaveAsObject is set to No for most objects on the screen. Error -255 will be returned upon an attempt to save such a screen. In this situation, change SaveAsObject to Yes for the objects that contain large amounts of text (often Program, Textbox and Flextext objects), and save the screen again.

Also see — GOSUB

SCALE Command

Applies to A-pex3 Xgraphics programming

Description Allows experienced authors to redefine the X-Y coordinate system of a window.

Syntax SCALE (Mode [, X1, Y1, X2, Y2])

Details Mode is either 0 or 3. Use 0 to redefine the coordinates; include the X1, Y1, X2 and Y2 parameters. Use 3 alone to reset the coordinate system back to the default.

X1 and Y1 are numeric values in your custom coordinate system to assign the upper-left corner of the inside of the window.

X2 and Y2 are numeric values in your custom coodinate system to assign the lower-right corner of the inside of the window.

When Mode is greater than 0, SCALE also proportionally resizes the objects in the window to match the new coordinate system.

Example The following example draws a box that occupies the left half of the window, regardless of the size of the window:

```
SCALE (0, 0, 0, 100, 100)
FBOX (0, 0, 50, 100, 14)
SCALE (3)
```

Notes SCALE influences the location of objects as well as Xgraphics.

Be sure to include a space between SCALE and (.

Also see AutoResize, Scrollable

Scn() Function

Applies to A-pex3 programming

Description Returns a number greater than zero if ScreenName exists in the currently loaded screen library; otherwise it returns 0.

Syntax scn(ScreenName)

Details Many authors employ this function to check if a screen exists before attempting to branch to it. This feature is particularly useful when the screen name is being generated through user action or random numbers.

Example The following example generates a random number, checks if the screen exists, and branches to it if it does:

```
quizpick = "q" + rnd(11)
IF scn(quizpick) > 0 THEN
  BRANCH {quizpick}
ELSE
  BRANCH endquiz
ENDIF
```

Scrollable Attribute

Applies to	Layout object
Description	Controls whether scroll bars are displayed within the window when VirtualHeight is greater than the height of the inside of the window, and/or VirtualWidth is greater than width of the inside of the window.
Settings	Yes display scroll bars when needed No do not display scroll bars
Details	This nifty feature lets the user scroll the contents of the window when the window is too small to see all its objects at once. You determine the size of the scrollable area via the VirtualWidth and VirtualHeight attributes.
Example	Some authors use this feature to display a large BgndPicture in the window (for example, 1024 x 768 pixels in size). If the user's computer can't display such a large picture (perhaps the limit is 640 x 480 pixels), the Scrollable feature lets the user pan left/right and up/down as needed.
Also see	AutoResize, VirtualHeight, VirtualWidth, WindowState

ScrollBars Attribute

Applies to	Listbox, Textbox objects
Description	Controls whether scroll bars are displayed within the object.
Settings	0 none 1 horizontal (vertical for Listbox) 2 vertical (horizontal for Listbox) 3 both bars
Details	Even when no scroll bars are present, the user can scroll text via the cursor keys. For a Listbox, the scrollbars do not appear unless the items are of sufficient length to require them.
Also see	MultiLine

Search and Replace Window

The Search and Replace Window is accessible via the main Author window's Utilities pull-down menu. The Search and Replace utility helps you find and/or change text throughout a book.

SEARCH vs. SEARCH AND REPLACE

To search without replacing, enter the search text, but leave the replacement field empty. To also replace the text, enter something in the replace field.

ITEMS TO SCAN

These features let you restrict the search to certain object classes and attributes. For example, to search Textbox objects only, un-check the "All objects" check box, then highlight Textbox in the list below.

Attributes (* = all) - Via this field you can restrict the scan to certain attributes. To scan only one attribute, enter its name. To scan multiple attributes, enter the attribute names separated by commas. To scan all attributes, either leave the field emptyor enter an asterisk (*).

SEARCH OPTIONS

Text Anywhere - Enable this to scan all attributes that are stored as text. Note that the contents of Program objects are considered text.

Numeric Attributes - Enable this to scan all attributes stored as numbers for numeric equivalence. So, a search for 5 will match both 5 and 5.0; note that -5 will not match, provided you disable the Text Anywhere option above.

Match Case - Enable this if you only want to match text that has the same upper and lower case letters as you specify in the search field.

Match Whole Word Only - Enable this to match only complete words; matches within words will be ignored.

Verify Interactively - Enable this to make Everest alert you to each match and ask if you want to open the page for editing. When disabled, Everest will simply count the number of matches.

Display Page Contents - Enable this to have Everest show you the pages are they are scanned. This can make the search run more slowly.

Sel() Function

Applies to A-pex3 programming

Description Returns a unique random number.

Syntax sel(VarName)

Details The Sel() function returns a randomly chosen unique integer between 1 and the length of the contents of VarName. Sel() is very handy for generating numbers for quiz question selection purposes.

Before calling the Sel() function, initialize VarName to one or more blank spaces. The number of blank spaces should equal the largest number you want Sel() to return.

When you use Sel(), it generates a random number N, and places the character "Y" in the VarName variable at the Nth character position to flag that number as "used."

If all the numbers have been generated (i.e. VarName contains no more blank spaces), Sel() returns 0. If VarName is empty (i.e. a null string), Sel() returns -1.

Example A common use for random numbers is to extract different exam questions in a random order from a pool of available questions. Here's how to employ Sel() to help you choose 20 of 100 questions:

1) Create a pool of question pages with names consisting of a letter plus a number. For example, name the first page q1, the second q2, etc. Put one question on each page.

2) Design the branching on each question page so that it returns to a main page after the question. You could name the main page "selectq."

3) Create an exam initialization page. You could name is "initexam." Design it to branch to selectq. Later, when ready, start the exam by running the initexam page.

4) Add a Program object to the initexam page, and enter the following calculations:

```
flagvar = 32$100        $$ 100 spaces
counter = 0             $$ init question counter
BRANCH selectq
```

5) On the selectq page, use a Program object to check if the question counter variable has reached the maximum you want (for example, 20 questions), and if it has, end the exam (branch to the endexam page, which you need to create separately). Otherwise, use the Sel() function to generate a unique random number, and concatenate the number to make a page name:

```
IF counter = 20 THEN    $$ if exam is done
  BRANCH endexam
ELSE
  counter++             $$ increment counter (+1)
  nextq = "q" + sel(flagvar)
  BRANCH {nextq}
```

```
ENDIF
```

Notes — When debugging, the series of random numbers will always be the same each time you run the AUTHOR program. In ERUN, the random numbers are different.

Also see — Rnd() Function, Scn() Function, Sfl() Function

SelLength Attribute

Applies to	Input, Textbox objects
Description	Retrieves or sets the number of characters that are highlighted in the object.
Also see	AllowSelection, SelStart, SelText

SelStart Attribute

Applies to	Input, Textbox objects
Description	Retrieves or sets the starting location of highlighted text in an object.
Also see	AllowSelection, SelLength, SelText

SelText Attribute

Applies to	Input, Textbox objects
Description	Retrieves or changes the highlighted text in an object.
Also see	AllowSelection, SelLength, SelStart

ServerClass Attribute

Applies to	OLE object
Description	Determines the server class for later queries into the registration database.
Details	Click in the Attributes window for more information to see a list of ServerClasses available on your computer.

ServerShow Attribute

Applies to	OLE object	
Description	Determines whether Everest tells the server application to display itself upon Action 7 (activate).	
Settings	Yes	tell server to display itself
	No	do not tell server to display itself
Also see	Action, Focus	

ServerType Attribute

Applies to	OLE object	
Description	Determines the type of link between the client and server.	
Settings	0	Linked. All the object's data is managed by the server.
	1	Embedded. All the object's data is managed inside your project.
	2	Static. No data is saved.
Also see	Action	

SetFocus Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Gauge, HScroll, Input, Listbox, Mask, OLE, Option, Picture, SPicture, VScroll objects

Description — Sets the focus (cursor) to the desired object. Write-only.

Settings

-1	move focus to the object
0	do nothing
1	move focus to the object and do pending events

Details — The object with "focus" is the one that is highlighted or awaiting user input. For example, if a page has several Input objects, the one with the cursor is the one that has the focus.

Authors typically use SetFocus to force the cursor to a particular field. Only objects that are enabled and visible (Enabled and Visible attributes set to -1) can receive the focus.

SetFocus does not appear in the Attributes window. It is accessible only via A-pex3 programming. If you want the cursor to appear in a particular object when the user first encounters a page, set that object's Initially attribute to 7 or 15.

Example — The following A-pex3 code example moves the focus to an Input field with IDNumber 3:

```
Input(3).SetFocus = -1
```

The following A-pex3 code example moves the focus to a Button with IDNumber 2, and interrupts Everest to allow pending events to be performed:

```
Button(2).SetFocus = 1
```

Note — This is a write-only attribute. The Obj() function can tell you which object has the focus currently.

Due to a bug in Windows, do not set SetFocus to 1 while processing a ChangeEvent.

Also see — Enabled, GotFocusEvent, Initially, LostFocusEvent, Obj() Function, TabStop

Settings Window

The Settings Window is accessible via the main Author window's File menu. The Settings Window lets you configure default values for various items. These settings are saved in the EVEREST.INI file.

INTERNET/INTRANET

Cache Path - This is the disk path to the location of the Inter/intranet cache on the local computer. When Everest downloads files from the Inter/intranet, it stores them in a location known as the cache. Any existing subdirectory on your hard drive (or RAM drive) can serve as the cache. The size of the cache must be sufficient to hold the total of all files that might get downloaded during the session. When you exit the Everest program, it automatically deletes any files it put in the cache. Example: `C:\cache`

Default Site for ~: Drive - Enter the address of the Web site that contains the source of Inter/intranet files referenced via the "~:" drive wildcard. For example, for an SPictureFile attribute, if you enter "~:cat.pcx", Everest will automatically attempt to download the file from the Web site you specify in this field. Example: `http://www.insystem.com`

Timeout - This is the number of seconds to wait for a reply from the Inter/intranet. If Everest sends a message (such as a request to download a file) to the Inter/intranet, and does not receive a reply within the time period you specify here, Everest will quit waiting and return an error message (error -317, Socket timeout). Example: `20`

FTP Upload Server - When uploading files to the Inter/intranet, this is the name of the server you want Everest to use. Contact your ISP or site manager to determine your server's name. At run time, the setting you specify is held in Sysvar(183). Example: `ftp.insystem.com`

FTP Upload Directory - When uploading files to the Inter/intranet, this is the name of the directory in which you want to store the files. Contact your ISP or site manager to determine the name on your system. Held in Sysvar(184). Example: `/xyz/www/html/insystem`

Equivalent URL - Enter the Inter/intranet URL that is equivalent to the FTP Upload Directory you specified in the field above. This information helps Everest determine the appropriate (sub)directory to which to upload the file. Held in Sysvar(188). Example: `http://www.insystem.com`. To illustrate, if the examples above for upload directory and URL are equivalent, then if you run a book from `http://www.insystem.com/evdemo`, and edit that book on the fly, Everest will upload the changes to the `/xyz/www/html/insystem/evdemo` directory.

FTP Upload User Name - When uploading files to the Inter/intranet, this is the identification to use for log on purposes. Most FTP sites require some sort of name before they will accept uploaded files. Contact your ISP or site manager to determine the correct name for your system. Held in Sysvar(185). Example: `intersys`

FTP Upload Password - When uploading files to the Inter/intranet, this is the password that accompanies the user name for log on purposes. If you make an entry here, Everest will store it in the EVEREST.INI file (an unsecure location). If you prefer, leave this field empty, and Everest will prompt you to enter the upload password just prior to uploading a file. Contact your ISP or site manager to determine the correct password for your system. Held in Sysvar(186). Example: `opensesame`

CacheOption - Choose one of the settings from the list provided. This setting controls how Everest

handles the cache, and whether .ZIP is the preferred format for uploading and downloading files. CacheOption controls four things at once, and can have a value of 0 to 15. You add the values from the list below to determine the proper setting:

1 Indicates most of your project files will be in .ZIP form, rather than in their uncompressed, original form. ERUN will look for a .ZIP file first. Also influences the storage format (normal or ZIPped) of uploads after an edit-on-the-fly.

2 If the uncompressed, original file cannot be found for downloading, do not also check for the same file in .ZIP form; instead, report that the file cannot be found.

4 Do NOT clean up the cache when done; leave downloaded files in the cache.

8 Check the cache first before downloading. If the required file already exists in the cache (such as from a download during a prior session, or from an initial installation), then use it rather than downloading another copy.

SNAP-TO GRID

Alignment - You can tell Everest to automatically align objects on an imaginary grid in the VisualPage editor. Choose one of the options in the list.

Horizontal Spacing - This is the X-direction spacing (in pixels) between grid lines. Example: `10`

Vertical Spacing - This is the Y-direction spacing (in pixels) between grid lines. Example: `10`

VISUALPAGE SIZE GUIDES

Width (X pixels) - While authoring, the VisualPage editor can be expanded to fit the size of your monitor. However, often this is larger than the visible area of the page on the end user's computer. The Size Guides act as a reminder of the size of the window on the end user's computer. Enter the desired value. Example: `635`

Height (Y pixels) - See above. Enter the desired value. Example: `455`

ATTRIBUTE DISPLAY ORDER

The Attributes window can list object attributes in either Functional or Alphabetical order. Choose the approach you prefer. (Alphabetical order is somewhat slower.)

AUTHOR'S NAME

Enter your name here. When you edit a page, Everest saves this name with it. This information can later prove helpful for multi-author projects.

SAVE .INI UPON QUIT

Enable this feature to automatically save the EVEREST.INI when you exit the AUTHOR program; doing so saves your current settings, window positions, etc., which is handy for the next time you start the program.

Sfl() Function

Applies to	A-pex3 programming
Description	Randomly shuffles one or more attributes among a group of objects of a particular class.
Syntax	sfl(Class, StartID, EndID, Attribute [,Attribute2 ...])
Details	For CBT projects, authors often use this function to swap the arrangement of choices for a multiple choice question page. The objects whose attributes are to be shuffled must have consecutive IDNumbers (in the range StartID to EndID). No useful information is returned directly by the function.
Example	The following A-pex3 programming example shuffles two attributes of four Button objects with IDNumbers from 11 to 14: `dummyvar = sfl("Button", 11, 14, "Caption", "Answers1")`
Notes	Up to five attributes can be shuffled at once. Be sure to surround the Class and Attribute names with quotes.
Also see	Rnd() Function

ShadowColor Attribute

Applies to	Button, Check, Combo, Frame, Gauge, Listbox, Option objects
Description	Specifies the color to employ as the dark color for 3-D shadowing effects.
Double click	Opens color dialog box. Click on the color of your choice.
Details	Refer to the BackColor attribute.
Also see	LightColor

Shape Attribute

Applies to	Shape object	
Description	Controls the appearance of a Shape object.	
Settings	0	rectangle
	1	square
	2	ellipse
	3	circle
	4	rounded-corner rectangle
	5	rounded-corner square
Also see	Shape Object	

Shape Object

Description The Shape object draws a simple shape on the page, such as a circle or square.

Attributes
BorderColor
BorderWidth
Bottom
ClickEvent
Comment
Condition
Create
Destroy
DrawMode
FillColor
FillStyle
Height
IDNumber
Initially
Left
MouseLeaveEvent
MouseOverEvent
MouseStayEvent
Move
Name
OutlineStyle
Right
SaveAsObject
Shape
ShapePointer
Top
Visible
Width
ZOrder

Details Use a Shape object when you want to quickly and easily draw something on the window. Refer to the Shape attribute for a list of shapes available.

If you have many shapes to draw from one page, you should employ A-pex3 Xgraphics drawing commands (such as CIRCLE and BOX) instead because Windows handles them more efficiently.

Everest draws Shape object graphics in a layer beneath other objects, but above Xgraphics vector graphics and the BgndPicture image. Since the interior of Shapes can be made transparent (FillStyle = 1), they are ideal for detecting user clicks on parts of a BgndPicture. Shapes also have a unique feature in that Everest processes their ShapePointers, MouseOverEvents and ClickEvents first (i.e. even if another object is on top of them), making them ideal for "hot spots" on Pictures, etc. Refer to the ClickEvent attribute for additional information

Also see BgndPicture, BOX, CIRCLE, FBOX, Line object, RBOX, Shape

ShapePointer Attribute

Applies to Shape object

Description Controls the appearance of the mouse cursor while it is positioned over the Shape at run time.

Settings

0	default
1	arrow
2	cross-hairs
3	I-beam
4	icon
5	N, S, E, W arrows
6	NE, SW arrows
7	N, S arrows
8	NW, SE arrows
9	W, E arrows
10	up arrow
11	hourglass
12	"not allowed" symbol

Details The Shape has the highest priority for mouse cursor appearance (i.e. even if another object is on top of the Shape, the Shape's mouse cursor setting will be used).

If you do not enter a ShapePointer, or if the Shape's Visible attribute is 0, Everest processes the mouse cursor as if the Shape were not present at all. That is, the MousePointer for an obscuring object (or for the window) will be employed. If you want the Shape to be invisible, but still employ the ShapePointer, set the Shape's OutlineStyle to 0 (transparent) and its FillStyle to 1 (transparent).

Also see MousePointer

Shl() Function

Applies to A-pex3 programming

Description Invokes an external program, and/or transfers the focus to it.

Syntax shl(Name [, WindowStyle] [,Wait])

Name is a character string that contains the name of the program.

WindowStyle is a number that specifies the appearance of the window in which the external program initially appears.

Wait is a flag that tells Everest whether to wait for the external program to finish executing.

Details WindowStyle should be one of the following values:

1	normal with focus (default, if omitted)
2	minimized with focus
3	maximized with focus
4	normal without focus
6	minimized without focus
-1	directs focus to a running program, where Name is the name in the title bar of the program

Wait should be one of the following values:

0	continue executing the Everest project (default, if omitted)
1	do not continue until the external program is finished

If successful, Shl() returns a non-zero number. If an error occurs, Shl() returns 0, and places a numeric error code in Sysvar(1).

Examples The following commands start the Windows paint brush application and check for an error:

```
ecode = shl("pbrush.exe")
IF ecode = 0 THEN
  ecode = "Error #" + sysvar(1) + " occurred!"
  dummyvar = mbx(ecode)
ENDIF
```

This example executes a Web browser program in order to access a site on the Internet (all on one line). Notice how the site address is passed as a command line parameter:

```
ecode = shl("C:\netscape\netscape
http://ourworld.compuserve.com/homepages/intersys")
```

Also see Dde() Function, OLE Object

ShowButtons Attribute

Applies to — Media object

Description — Determines whether media device control buttons (play, pause, fast forward, etc.) are displayed on the page at run time.

Settings

Yes	display buttons
No	do not display buttons
> 0	(run time only) to set visibility status of individual buttons (see Details)

Details — Display the buttons if you want the user to be able to control the media device manually. Everest automatically enables only those buttons that are available for the device.

At run time, via programming, you can control the visibility of the individual buttons. To do so, set ShowButtons equal to a number that is the sum of the values in the table below:

back	1
eject	2
next	4
pause	8
play	16
prev	32
record	64
step	128
stop	256

If no buttons are displayed, the user will not be able to operate the device manually; consequently, you would then need to control the device via the Media object's Commands or the Mci() function.

Example — The following A-pex3 programming example makes only the Pause, Play and Stop buttons visible for the Media object with IDNumber 1:

```
Media(1).ShowButtons = 280    $$ 280 = 8+16+256
```

Also see — Command, DeviceType, Mci() Function, Orientation

Silent Attribute

Applies to	Media object	
Description	Determines whether sound (if available) plays audibly.	
Settings	Yes	turn off sound
	No	play sound

Sin() Function

Applies to	A-pex3 programming
Description	Returns the trigonometric sine of an angle.
Syntax	sin(Angle)
Details	Express Angle in radians. To convert from degrees to radians, multiply by (pi/180).
Example	The following example stores the sine of a 45 degree angle in the variable named myangle:

```
myangle = sin(45 * 3.141593 / 180)
```

Also see Atn() Function, Cos() Function, Tan() Function

Sorted Attribute

Applies to	Combo, ListBox objects
Description	Determines whether the items in the list are displayed in alphabetical order.
Settings	Yes display in alphabetical order No display in natural order
Also see	ItemList, Srt() Function, Style

SoundDevice Attribute

Applies to	Animate object	
Description	Determines which media device is used as the source of sounds during the animation.	
Settings	CDAudio	redbook audio from CD-ROM
	Sequencer	MIDI audio
	Videodisc	audio from laserdisc player
	WaveAudio	digitized sound
Also see	SoundFile	

SoundFile Attribute

Applies to	Animate object
Description	Specifies the name of the file that contains audio to be played during the animation, or the commands to send to the SoundDevice.
Double click	Opens file dialog box. Double click on the file you want.
Details	If SoundDevice is "Sequencer" or "WaveAudio" this attribute specifies the name of the audio file to play. If SoundDevice is "CDAudio" or "Videodisc" this attribute specifies the MCI command string to send to the device. Consult a Windows Multimedia guide for appropriate command strings.
Also see	Animate Object, SoundDevice

SourceDoc Attribute

Applies to	OLE object
Description	Determines the name of the file to use when you create an object from a file (Action = 1).
Also see	Action, SourceItem

SourceItem Attribute

Applies to	OLE object
Description	Determines the data within a file to be linked.
Details	ServerType must be set to 0 (Linked) for this to work, and SourceDoc must specify the name of the file. Consult the server application's documentation to learn what SourceItems it allows.
Also see	Action, ServerType, SourceDoc

Special Object

Description — The Special object is obsolete. Use a Program object instead. In prior versions, the Special object could hold up to 8 lines of A-pex3 programming code.

Attributes
Comment
Condition
Name
Special1...Special8

Details — The functionality of Special objects was duplicated by the more flexible Program objects. Use a Program object where you would have used a Special. For upward compatibility, this version of Everest supports Special objects in older projects.

Also see — Special1

Special1, Special2, Special3, Special4, Special5, Special6, Special7, Special8 Attributes

Applies to Special object

Description Each Special attribute can contain one line of A-pex3 programming code.

Details The Special object is ideal when you need to perform a simple A-pex3 command or a very short program.

The Special object is stored entirely within the page; a separate object is NOT stored on the disk. Special objects cannot be accessed via Object Instancing.

The Special object is now obsolete, and should be replaced in your projects with a Program object, which allows larger programs and can be accessed via GOSUB.

Example To increment the value stored in the variable named counter, you would enter either of the following expressions for the Special1 attribute:

```
counter = counter + 1
```

or

```
counter++
```

Also see SaveAsObject

SpecialEffect Attribute

Applies to Layout, Picture objects

Description Specifies the effect (slide, overlay, etc.) with which to display the PictureFile or BgndPicture image.

Settings

0	display normally (no special effect)
1	explode
2	horizontal curtain close
3	horizontal curtain open
4	horizontal rotate
5	implode current image
6	slide down
7	slide right
8	slide left
9	slide up
10	push down
11	push left
12	push right
13	push up
14	vertical curtain close
15	vertical curtain open
16	vertical rotate
17	simply scale image to size of Picture box or window at run time
18	overlay dissolve randomly
19	overlay down
20	overlay right
21	overlay left
22	overlay up
23	overlay zig zag
24	overlay center out
25	overlay into center
26	overlay smear down
27	overlay NW to SE

Details SpecialEffect works via the Windows BitBlt function, which, in turn, relies heavily on the stability of the driver software for your particular video display adapter. Problems with SpecialEffect (hangs, GPFs, partial images) are typically caused by bugs in the video display driver for your computer. You can often work around these problems by running Windows in a different video resolution and/or color depth. Available memory can also impact the success of SpecialEffect; smaller images with fewer colors work best. If you continue to experience problems, obtain a driver update for your video display adapter.

Setting the size of the Picture object to match that of the image can improve the visual quality and speed of the effect. Enable AutoSize to automatically resize the object as needed.

SpecialEffect has the side effect of scaling the image to fit the Picture or window. To

avoid scaling a BgndPicture, add 1000 to the SpecialEffect number.

Experiment to find an acceptable effect for your project. Due to video driver instability, we must ask that you use SpecialEffect at your own risk.

Notes

If your SpecialEffect is not visible at run time, make sure LockUpdate is set to No.

Changing the value of SpecialEffect of a Picture at run time via A-pex3 programming causes the current image to replot with the new effect. If you wish to avoid this replot, use a negative value (for example, -18 instead of 18). Authors often use this technique to defer display of the SpecialEffect until the next image is loaded (via the PictureFile attribute).

By default, Everest will display the SpecialEffect in a minimum of approximately 1 second (i.e. on an infinitely fast computer, the effect would require 1 second to run to completion). Everest automatically adjusts to the speed of the computer. To change the duration of the SpecialEffect, set Sysvar(162) at run time via A-pex3 programming to the desired minimum time.

The granularity (block size) of the overlay style SpecialEffects can be modified. To do so, set Sysvar(163) to the desired number of blocks per line (20 is the default).

Also see

CopyBgnd, CopyPic, DrawText, Picture Object, SPicture Object, STYLE, Tile

SPicture Object

Description With the SPicture object you can include sizable bitmapped pictures on your page.

Attributes
AnimPath
BackColor
Bottom
ClickEvent
Comment
Condition
Create
DblClickEvent
Destroy
DragMode
Enabled
Height
IDNumber
Initially
Left
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
Name
Right
SaveAsObject
SetFocus
SPictureFile
Top
Update
Visible
Width
Zev
ZOrder

Details The SPicture object can display .BMP, .GIF, .JPG, .PCX, .RLE, .TGA and .TIF picture files containing up to 16.7 million colors. It scales the SPictureFile image to fit into the box.

Similar functionality can be obtained with a Picture object with a SpecialEffect.

Also see Picture Object

SPictureFile Attribute

Applies to	SPicture object
Description	Specifies the name of the disk file that contains the picture to display in the SPicture object
Double click	Opens file dialog box. Double click on the file you want.
Details	Enter the name of the .BMP, .GIF, .JPG, .PCX, .RLE, .TGA or .TIF file to display. The SPicture object can handle bitmaps containing up to 16.7 million colors. For help with file locations, refer to Appendix F.
Also see	PictureFile

Sqr() Function

Applies to A-pex3 programming

Description Returns the square root of a number.

Syntax sqr(Numeric)

Example The following A-pex3 example computes the distance from the upper left corner of the window to the mouse cursor:

```
hdist = window(0).left - mse(1)
vdist = window(0).top - mse(2)
dist = sqr((hdist ^ 2) + (vdist ^ 2))
```

Also see Operators

Srt() Function

Applies to	A-pex3 programming
Description	Sorts the contents of an array.
Syntax	srt(Arrayname(Element))
Details	Arrayname is the name of the array to sort. The Srt() function sorts array elements 1 to Element into ascending order. If Arrayname(Element) is a number, Srt() performs a numeric sort. Otherwise, Srt() performs an alphabetic sort.
Example	`dummyvar = srt(names(60))`
Also see	Lod() Function, Sum() Function

StartAt Attribute

Applies to Media object

Description Specifies the location at which to start playing or recording a multimedia sequence.

Double click Opens the multimedia peek window.

Details This attribute is typically used to control the start point of a motion video clip, or the start point of CDAudio. When combined with the EndAt attribute, you can define the portion of the multimedia element to play.

You should express StartAt in the current TimeFormat. You can type the value of StartAt (if you know it), or peek at the media and have Everest generate the value for you.

MULTIMEDIA PEEK

To peek at the media, double-click on the StartAt line in the Attributes window. The Multimedia Peek window will appear with a row of buttons (Play, Stop, etc.). Everest activates only those buttons that are appropriate for the device. If all buttons are disabled, the DeviceType does not support peeking.

As the media plays, the Multimedia Peek window displays the current location in two formats: 1) as a 4-byte long integer Position, and 2) in the current TimeFormat as 4 individual values between 0 and 255, one for each byte, separated by colons.

The location values should be updated 10 times per second. We have found that some brands of multimedia hardware do not update the values this frequently. If you observe this problem, check with the hardware manufacturer to determine if you have the most current drivers.

When the media reaches the desired location, click either the "Use Position" or "Use TimeFormat" buttons. This copies the current corresponding location information to the StartAt attribute.

START AT BEGINNING

To play from the beginning of the media, leave the StartAt attribute empty.

Also see DeviceType, EndAt, TimeFormat

State Attribute

Applies to Button, Check, Option objects

Description Sets the status of the object.

Settings **BUTTON OBJECT**

0	up
-1	down

CHECK OBJECT

0	unchecked
1	checked
2	grayed

OPTION OBJECT

0	not selected
-1	selected

Details Care must be taken when changing State of an object at run time via A-pex3 programming because the object's ClickEvent fires. If the ClickEvent matches an xxxActivator in the WaiT object, execution jumps immediately to that Wait object. If the ClickEvent matches a global hot key page, that page is executed. Refer to the example below for more information.

Example If you wish to ignore the ClickEvent when changing the State of an object via A-pex3 programming, add 10 to the setting. For example, to depress the Button with IDNumber 1, and ignore its ClickEvent use:

```
Button(1).State = 9
```

Also see Value

Step Attribute

Applies to	HScroll, VScroll objects
Description	Sets the amount by which to change a scroll object's value when the user clicks on the scroll arrow.
Also see	LargeStep

STEP Command

Applies to A-pex3 programming

Description Engages or disengages the Debug window's step mode feature while debugging.

Syntax STEP (Action)

Details Use STEP anywhere you would an A-pex3 programming command. The action of the STEP command depends on the value of the Action parameter:

Action	Result
-1	STEP command is ignored
0	disengages the step mode feature
1	engages the step mode feature only if the Debug window is already open
2	opens the Debug window (if it is closed) and engages its step mode feature

Notes Everest ignores the STEP command at user run time.

For proper operation, include a space between the STEP command and the (.

Also see DPRINT

Stf() Function

Applies to	A-pex3 programming
Description	Presses keys in the current window, as if the user had typed them.
Example	The following A-pex3 example types the characters "answer1", presses Tab and types the characters "answer2"

```
dummyvar = stf("answer1{TAB}answer2")
```

Also see Key() Function

Style Attribute

Applies to	Combo object	
Description	Controls the appearance of the drop down list of items.	
Settings	0	the user can select from a drop down list or type in the editing area
	1	same as 0, except the drop down list is always displayed
	2	same as 0, except the user cannot type in the editing area (and the Text and Preset attributes become read-only)
Also see	MaxDrop	

STYLE Command

Applies to A-pex3 Xgraphics programming

Description Sets the various appearance attributes of subsequently drawn Xgraphics. Can also enable drawing on a hidden canvas for later display with a special effect.

Syntax STYLE (Item, Value)

Details The STYLE command controls the size and drawing style of the Xgraphics "pen." Select the attribute you want via the Item parameter; control the setting via the Value parameter.

Item	Possible Values
1	width of the pen; specify Value in pixels
2	set pen drawing mode; use one of the following Values: 1 = blackness 2 = not merge pen 3 = mask not pen 4 = not copy pen 5 = mask not pen 6 = invert 7 = xor pen (see Notes below) 8 = not mask pen 9 = mask pen 10 = not xor pen 11 = no drawing 12 = merge not pen 13 = copy pen (default & most common) 14 = merge pen not 15 = merge pen 16 = whiteness
3	set pen style; use one of the following Values (for non-solid drawings, the pen width must be 1): 0 = solid (default) 1 = dash 2 = dot 3 = dash-dot 4 = dash-dot-dot 5 = no drawing 6 = inside solid
4	set FillStyle (paint); use one of the following Values: 0 = solid 1 = transparent (default) 2 = horizontal lines 3 = vertical lines 4 = NW SE diagonal lines

5 = SW NE diagonal lines
6 = cross pattern
7 = diagonal cross pattern

-1 prepare for special effect; tells Everest to draw subsequent Xgraphics onto a hidden canvas; set Value to the IDNumber of the desired Picture object for output when you later use Item setting 2, or to 0 for the window background

-2 copy the hidden canvas image to the window background or Picture object chosen via Item setting -1, and employ the SpecialEffect indicated in the ValuE parameter

Examples The following example draws a dashed line in palette color 14:

```
STYLE (1, 1)            $$ width must be 1 for dashes
STYLE (3, 1)            $$ prep for dashes
LINE (0, 0, 100, 100, 14)
```

The following example fills a hidden canvas with a graduated color, then displays the result in the Picture object with IDNumber 3 using SpecialEffect 18 (dissolve):

```
STYLE (-1, 3)           $$ enable hidden; later show on 3
GFILL (0, 0, 0, 0, 0, 255)    $$ blue gradiant fill
$$ other Xgraphics commands could go here
STYLE (-2, 18)          $$ dissolve onto Picture(3) now
```

Notes For proper operation, include a space between STYLE and (.

Due to a bug in Windows, STYLE (2, 7) has been known to produce invisible output on certain computers.

After using STYLE (-1,... be sure to later use STYLE (-2,..., otherwise Everest will continue to draw Xgraphics on the hidden canvas. Everest automatically changes the value in Sysvar(108) during the operation of this feature,

Also see DrawText

Sum() Function

Applies to A-pex3 object

Description Returns the sum of the contents of an author defined array.

Syntax sum(Arrayname(Element) [, Separator])

Details This function adds together the contents of the elements of Arrayname from 1 to Element. If Arrayname(Element) is a string, or if the Separator parameter is included, Sum() performs string concatenation. Otherwise, Sum() performs numeric addition.

Example The following A-pex3 example calculates the average of the values stored in the array named scores:

```
elements = arr("scores")
avg = sum(scores(elements)) / elements
```

The following example concatenates the elements of the array named files, separating each with a semicolon:

```
elements = arr("files")
all = sum(files(elements), ";")
```

Also see Lod() Function, Srt() Function

SystemModal Attribute

Applies to Layout object

Description Determines whether the user cannot activate any other windows.

Settings
Yes user cannot activate any other window
No user can activate other windows (by clicking on them)

Details SystemModal is useful when you want to prevent the user from leaving your project. When you enable SystemModal for a window, that window becomes the only one available for use (i.e. Microsoft Windows will ignore the user's attempts to activate another window by clicking on it).

If you enable SystemModal, make sure you provide a method for a user to exit your project, otherwise they will be forced to reboot the computer (causing possible data loss).

Example The following A-pex3 code makes the current window system modal:

```
Window(sysvar(8)).SystemModal = "Yes"
```

Notes To help prevent you from locking up your computer while editing and test running your project, SystemModal only takes effect at user run time.

Also see Mbx() Function, WindowState

Sysvar() Variables

Applies to A-pex3 programming

Description The elements of the Sysvar() array contain the values of system variables.

Details System variables contain information that is sometimes useful in your programs. System variables are read-only, unless otherwise noted below. NOTE: Changing the value of a read-only system variable can produce unpredictable results!

sysvar(0)	number of system variables (# elements in sysvar() array)
sysvar(1)	numeric code of last error
sysvar(2)	screen twips per pixel x
sysvar(3)	screen twips per pixel y
sysvar(4)	# questions still active
sysvar(5)	# questions answered correctly; continues to increment with each Judge object; set to 0 to reinitialize; also see sysvar(175)
sysvar(6)	# questions attempted; continues to increment with each Judge object; set to 0 to reinitialize; also see sysvar(176)
sysvar(7)	score: sysvar(5)/sysvar(6) * 100
sysvar(8)	active window number (-2 through 8)
sysvar(9)	last mouse click x location
sysvar(10)	last mouse click y location
sysvar(11)	last mouse button clicked (1 = left, 2 = right, 4 = middle, or sum for combinations)
sysvar(12)	keycode of last key pressed (see Appendix A)
sysvar(13)	key trapping flag
sysvar(14)	current nesting level of Include objects (0=none)
sysvar(15)	code number of Everest application currently running (-1 while editing, 0 = debug, 1 = user mode)
sysvar(16)	disk path to book file
sysvar(17)	book file name
sysvar(18)	current page name
sysvar(19)	width of display (pixels)
sysvar(20)	height of display (pixels)
sysvar(21)	lines to print per printer page
sysvar(22)	# lines printed since last auto-form feed
sysvar(23)	user quit flag (interrupt A-pex3 code)
sysvar(24)	code number of last active AUTHOR window
sysvar(25)	last page line loaded (used internally)
sysvar(26)	name of last author of currently loaded page
sysvar(27)	date of last edit of currently loaded page
sysvar(28)	name of author
sysvar(29)	PicBin loaded list
sysvar(30-45)	16-color palette colors
sysvar(46)	snap-to-grid horizontal spacing
sysvar(47)	snap-to-grid vertical spacing
sysvar(48)	snap-to-grid active while editing?
sysvar(49)	window relocated bits
sysvar(50)	attributes window sort style
sysvar(51)	save .INI upon quit

sysvar(52)	active window bits
sysvar(53)	allow paint event window bits
sysVar(54)	0 = previewing, 1 = running
sysvar(55)	book name lookup table
sysvar(56)	path to & name of user records file
sysvar(57)	path to & name of user comments file
sysvar(58)	CALL stack
sysvar(59)	skip program syntax check flag
sysvar(60)	copy of sysvar(52) upon edit
sysvar(61)	name of page running in window 1
sysvar(62)	name of page running in window 2
sysvar(63)	name of page running in window 3
sysvar(64)	name of page running in window 4
sysvar(65)	name of page running in window 5
sysvar(66)	name of page running in window 6
sysvar(67)	name of page running in window 7
sysvar(68)	name of page running in window 8
sysvar(69)	last active window during edit
sysvar(70)	backup stack window flag bits
sysvar(71)	backup stack for window 1
sysvar(72)	backup stack for window 2
sysvar(73)	backup stack for window 3
sysvar(74)	backup stack for window 4
sysvar(75)	backup stack for window 5
sysvar(76)	backup stack for window 6
sysvar(77)	backup stack for window 7
sysvar(78)	backup stack for window 8
sysvar(79)	window book pointer list
sysvar(80)	menu stack window flag bits
sysvar(81)	menu stack for window 1
sysvar(82)	menu stack for window 2
sysvar(83)	menu stack for window 3
sysvar(84)	menu stack for window 4
sysvar(85)	menu stack for window 5
sysvar(86)	menu stack for window 6
sysvar(87)	menu stack for window 7
sysvar(88)	menu stack for window 8
sysvar(89)	page pointer for preview
sysvar(90)	page pointer for peek
sysvar(91)	page pointer for VisualPage
sysvar(92)	page pointer for window 1
sysvar(93)	page pointer for window 2
sysvar(94)	page pointer for window 3
sysvar(95)	page pointer for window 4
sysvar(96)	page pointer for window 5
sysvar(97)	page pointer for window 6
sysvar(98)	page pointer for window 7
sysvar(99)	page pointer for window 8
sysvar(100)	disabled hot key list
sysvar(101)	window guide X
sysvar(102)	window guide Y

sysvar(103)	last BgndPicture loaded list
sysvar(104)	icon height in Book Editor
sysvar(105)	total questions answered correctly
sysvar(106)	total questions attempted
sysvar(107)	score: sysvar(105)/sysvar(106)*100
sysvar(108)	Picture object IDNumber for Xgraphics
sysvar(109)	last mouse down x location
sysvar(110)	last mouse down y location
sysvar(111)	class number of object on which another object was dropped (see table with Obj() function)
sysvar(112)	IDNumber of object on which another object was dropped
sysvar(113)	class number of last object that was dropped
sysvar(114)	IDNumber of last object that was dropped
sysvar(115)	run time FontSize scale factor
sysvar(116)	disk path to book being edited
sysvar(117)	file name of book being edited
sysvar(118)	name of page being edited
sysvar(119)	copy of sysvar(54) during edit on fly
sysvar(120)	interrupt number for ext(51)
sysvar(121)	AX register for ext(51)
sysvar(122)	BX register for ext(51)
sysvar(123)	CX register for ext(51)
sysvar(124)	DX register for ext(51)
sysvar(125)	[reserved]
sysvar(126)	non-zero when Refresh is being performed
sysvar(127)	active window bits when user logged off
sysvar(128)	name of page at which user would resume if an @restart page did not exist; use BRANCH {sysvar(128)} on @restart page to branch to it
sysvar(129)	password (for custom log on)
sysvar(130)	unique user number (recs in use)
sysvar(131)	user's first name
sysvar(132)	user's last name
sysvar(133)	user's group name
sysvar(134)	user's ID#
sysvar(135)	# times user has logged on
sysvar(136)	# times user has logged off
sysvar(137)	date/time/timer of record creation
sysvar(138)	date/time/timer of current log on
sysvar(139)	date/time/timer of last log off
sysvar(140)	total time logged on (in seconds)
sysvar(141)	save CMI flag
sysvar(142)	LogOn flag
sysvar(143)	object attributes array lookup table
sysvar(144)	number of window active when user logged off
sysvar(145)	upon user log on: -1=resuming, 0=not resuming
sysvar(146)	next page preload data
sysvar(147)	next page preload name
sysvar(148)	character location of Mask object validation error
sysvar(149)	StarPath value
sysvar(150)	External object reply flag

sysvar(151)	show Attribute descriptions flag
sysvar(152)	path for embedded temporary files
sysvar(153)	ignore Refresh counter
sysvar(154)	key and event queue
sysvar(155)	event queuing status
sysvar(156)	while between Wait objects at run time: -1=queue keys, 0=normal, 1=discard keys; can be modified
sysvar(157)	key queuing status
sysvar(158)	while between Wait objects at run time: -1=queue events, 0=normal, 1=discard events, can be modified
sysvar(159)	last mouse x location
sysvar(160)	last mouse y location
sysvar(161)	path to EVEREST.MSG file
sysvar(162)	SpecialEffect duration (in seconds; approximate minimum)
sysvar(163)	SpecialEffect granularity
sysvar(164)	path to EVEREST.INI file
sysvar(165)	computer graphics speed benchmark
sysvar(166)	computer computational speed benchmark
sysvar(167)	DDE reply timeout period (seconds)
sysvar(168)	[reserved]
sysvar(169)	counter for various operations
sysvar(170)	special keyboard configuration bits; add together: 1 = beep if user presses Enter in non-MultiLine Input object; 2 = disable Cut and Paste keys in Input objects; 4 = disable Alt+shortcut key handler; 8 = generate event upon key up (Everest adds 8000 to key event code)
sysvar(171)	SaveAsObject highlight color
sysvar(172)	window CloseEvent reason
sysvar(173)	MouseStayEvent duration
sysvar(174)	MouseStayEvent cancel
sysvar(175)	# questions answered correctly, as counted solely by the most recent Judge object executed
sysvar(176)	# questions attempted, as counted solely by the most recent Judge object executed
sysvar(177)	the number of the Listbox column most recently clicked (the leftmost column is 0)
sysvar(178)	Inter/intranet source address (use ~: as drive letter)
sysvar(179)	Local cache for Inter/intranet downloads
sysvar(180)	time of last receipt
sysvar(181)	timeout on Inter/intranet downloads, in seconds
sysvar(182)	Inter/intranet host port number (0 means use default)
sysvar(183)	FTP server name
sysvar(184)	FTP directory
sysvar(185)	FTP user name
sysvar(186)	FTP password
sysvar(187)	CacheOption file handling flags; add following values: 1=prefer .ZIP format for uploads/downloads; 2=don't perform auto .ZIP check; 4=do not delete cached files upon end of session; 8=check if file already exists (old copy) in cache location before downloading
sysvar(188)	equivalent URL for sysvar(184)

sysvar(189)	(reserved)
sysvar(190)	internal non-interrupt flag
sysvar(191)	CacheDate setting
sysvar(192)	[reserved]
sysvar(193)	[reserved]
sysvar(194)	[reserved]
sysvar(195)	file source for Internet Simulator
sysvar(196)	Internet Simulator data transfer chunk size
sysvar(197)	random name for user record uploads
sysvar(198)	used internally for data transfers
sysvar(199)	DrawShadow distance (in pixels)

Also see Ext() Function

TabOrder Attribute

Applies to — Button, Check, Combo, Flextext, Gauge, HScroll, Input, Listbox, Mask, Media, Option, Textbox, VScroll objects

Description — Controls the order in which the focus (highlight) moves from object to object in the window when the user presses the Tab key. Accessible only at run time via A-pex3 programming.

Details — By default, the tab order of objects is the order in which they were added to the window. Usually, this is the order in which they appear in the Book Editor. To alter this order, employ the TabOrder attribute.

Set TabOrder to a number that represents the desired order (lower numbers come first, starting with 0). Everest automatically renumbers the TabOrder of other objects to accommodate your changes.

Example — The following example makes the Input object with IDNumber 2 come first in the tab order:

```
Input(2).TabOrder = 0
```

Also see — TabStop

TabStop Attribute

Applies to	Button, Check, Combo, Flextext, HScroll, Input, Listbox, Mask, Media, Option, Textbox, VScroll objects
Description	Determines whether the user can move the focus to the object via the Tab key.
Settings	Yes user can Tab to object No Tab does not move focus to object
Also see	SetFocus, TabOrder

TagColorBgnd

Applies to	Combo, Listbox objects
Description	Determines the color behind the text for a selected item.
Details	Refer to the BackColor attribute.
Also see	TagColorFgnd

TagColorFgnd

Applies to	Combo, Listbox objects
Description	Determines the color of the text when an item is selected.
Details	Refer to the BackColor attribute.
Also see	TagColorBgnd

Tagged Attribute

Applies to Listbox object

Description Determines whether an item in a list is highlighted. Accessible at run time only.

Settings
0 item is not highlighted
-1 item is highlighted

Details Before using the Tagged attribute, always set the LookAt attribute to the number of the Item to reference. The first item in a list is number 0.

Example The following example concatenates the text of highlighted items of the Listbox with IDNumber 1 into the variable named all:

```
all = ""
pointer = 0: max = Listbox(1).ItemCount
DO IF pointer < max
  Listbox(1).LookAt = pointer
  IF Listbox(1).Tagged # 0 THEN
    all = all + Listbox(1).Item + "|"
  ENDIF
  pointer++
LOOP
```

Also see Item, LookAt, TaggedCount, TaggedList, TagStyle

TaggedCount Attribute

Applies to Listbox object

Description Returns the number of items that are currently highlighted in a Listbox. Read-only and available at run time only.

Example The following A-pex3 programming example uses TaggedCount to determine how many items are highlighted in the Listbox with IDNumber 1, then copies those items into an array:

```
REDIM chosen(Listbox(1).TaggedCount)
      max = Listbox(1).ItemCount
marked = Listbox(1).TaggedList
ptr = 1: count = 0
DO IF ptr <= max
  IF marked ^^ ptr = "X" THEN
    count++
    .LookAt = ptr - 1
    chosen(count) = .Item
  ENDIF
  ptr++
LOOP
```

Also see ItemCount, Tagged, TaggedList, TagStyle

TaggedList Attribute

Applies to Listbox object

Description Gets or sets the selection status of items in the list.

Details TaggedList is the quickest way to determine which items in a list are selected, or to set their status. Each character in the string returned by TaggedList represents one item in the list. The characters returned are either X (for selected items) or upper-case letter O (for non-selected items).

To set the selection status of the items in the list, set TaggedList to a string of X, O, T or - (hyphen) characters, up to one character for each item. The characters have the following meaning:

Character	Meaning
X	Select the item
O	Deselect the item
T	Toggle (reverse) the item
- (hyphen)	Do not change the selection status

When setting the selection status of items, if the Listbox contains more items than you specify in the string, Everest sets all remaining items in the Listbox according to the last character in the string.

Examples If the Listbox with IDNumber 1 contains 4 items, and the first and third are selected, the following returns "XOXO" in the variable named marked:

```
marked = Listbox(1).TaggedList
```

If the Listbox with IDNumber 1 contains 4 items, the following deselects the first, selects the second, and leaves the remaining items unchanged:

```
Listbox(1).TaggedList = "OX-"
```

The following toggles (reverses) all items in the Listbox with IDNumber 1:

```
Listbox(1).TaggedList = "T"
```

Notes Currently, TaggedList should not be used with Listboxes that have TagStyle set to 0 (single select). Use ItemIndex instead.

Assigning a string to TaggedList causes the Listbox's ClickEvent to fire for each item whose status changes. If you do not want the ClickEvent to fire, insert a | character (ASCII 124) at the start of the string.

Also see ItemList, Tagged, TaggedCount

TagStyle Attribute

Applies to	Listbox object
Description	Controls the number of items the user can highlight in the Listbox, as well as the approach to highlighting several items.
Settings	0 user can select only one item in list 1 user can select multiple items 2 user can select multiple contiguous items
Also see	Tagged, TaggedCount

Tan() Function

Applies to	A-pex3 programming
Description	Returns the trigonometric tangent of an angle.
Syntax	tan(Angle)
Details	Express Angle in radians. To convert from degrees to radians, multiply by (pi/180).
Example	The following example stores the tangent of a 45 degree angle in the variable named myarc:

```
myarc = tan(45 * 3.141593 / 180)
```

Also see Atn() Function, Cos() Function, Sin() Function

Technical Support

If you have a question about the proper use of Everest, please contact Intersystem Concepts as described below.

On-Demand Technical Support

If you do NOT have an active technical support contract, you can make use of our On-Demand technical support facilities. The fee (subject to change) is currently $60 per incident; add 10% outside the USA. Have your credit card (Visa or Master Card) ready, and call 888-8-AUTHOR, or 410-531-9000. If we determine that the trouble you experienced was due to a bug in Everest, the fee will be refunded in full.

Priority Author Support

If you have a Priority Author Support (PAS) contract, contact us for technical support in any of the following ways:

Call	410-531-9000
Fax	301-854-9426
E-mail	intersys@insystem.com

You can start or renew a PAS contract at any time. The fee (subject to change) is currently $600 per year per author (or $400 for 6 months; add 10% outside the USA), and includes free software updates and new versions for licensed Everest Suite users.

Other Support

Limited free technical support is available via mail. Send your written request to us at:

Technical Support
Intersystem Concepts, Inc.
P.O. Box 477
Fulton, MD 20759
USA

Due to the increasing number of Everest users, we cannot guarantee a response for free technical support within any particular time frame. However, your inquiry will be handled on a first-come, first-served basis as time permits.

Other Users

Consider sharing e-mail messages with other Everest users via the Internet in the comp.multimedia newsgroup.

Text Attribute

Applies to Combo, FlexText, Input, Mask, Textbox objects

Description Determines the text displayed in the object.

Examples For objects in which you can type text directly (such as a Textbox), you can embed variables and expressions in the text by surrounding them with { }. For example, to load and display the C:\CONFIG.SYS file at run time, enter the following as the text in a Textbox object:

```
{fyl(8, 1, "C:\config.sys")}
```

FLEXTEXT OBJECT

Flextext objects support several special embedded formatting and hypertext codes. At design time, you enter these codes as part of the text; at run time, Everest translates them into color, size, etc. For example, to italicize a word, surround it with \I and \i, as shown below:

```
To emphasize, use \Iitalics\i.
```

Here is a list of formatting codes:

\B \b	Bold. \B enables bold, and \b disables it.
\C \c	Color. \C enables an alternate color, and \c returns to the ForeColor. Express the desired alternate color via a hexadecimal value immediately after \C in the form BBGGRR. For example, for bright red text, use \ C0000FF. Dithered colors are translated into the nearest solid color.
\H \h	Heading. \H enables the character size expressed via the HeadingSize attribute, and \h return to FontSize. Should be placed on a line of its own (i.e. different sizes cannot be mixed on a given line).
\I \i	Italics. \I enabled italics, and \i disables italics.
\S \s	Strikethrough. \S enables strikethrough, and \s disables strikethrough.
\U \u	Underline. \U enables underline, and \u disables underline.

In addition to formatting codes, the Flextext object supports embedded hypertext style jump and popup words. You specify the event code that Everest should generate when the user clicks on such a word by including \K<event code>. This is best illustrated via examples. In the following example, the word Everest is a jump word, and generates event -100 when the user clicks on it:

```
This is the \J\K-100\kEverest\j authoring system.
```

In the example above, notice how the word Everest is surrounded by \J at the start and \j at the end. Also note that the event code is included immediately after the \J, and is

surrounded by \K and \k. At run time, the word Everest would be underlined with a solid line.

In the following example, Everest is a popup word:

```
This is the \P\K-100\kEverest\p authoring system.
```

The example above resembles the previous example, but instead of \J and \j, \P and \p are used to designate a popup. At run time, the word Everest would be underlined with a dashed line.

You can trap the jump and popup event codes as you do any other event code (such as that for the ClickEvent attribute) via a Wait object.

Everest also accepts any A-pex3 command (except CALL, OPEN and RETURN) between the \K and \k. This feature lets you perform BRANCHes, JUMPs and calculations without processing an event via a Wait object. In the following example, when the user clicks on the jump word "JumpPointer" in the sentence, the Hlp() function will invoke the Windows help system, and display the EVEREST.HLP topic for JumpPointer:

```
This is a \J\Kd=hlp("%:everest.hlp",261,"jumppointer")\
kJumpPointer\j cursor.
```

For a description of "%:" in the example above, refer to the Pth() function.

This example resembles the previous, except it executes a Web browser program via the Shl() function in order to access a site on the Internet:

```
Visit our Web site at \J\Kd=shl("C:\netscape\netscape
http://www.insystem.com")\khttp://www.insystem.com\j.
```

Notes

While editing the Text with the VisualPage editor, to enter a Tab character, press Alt+9 (use the number 9 on the numeric key pad).

To have Everest automatically generate the proper Flextext jump, popup or color code for you, first highlight the desired text, then choose the desired Flextext action from the main Author window's Edit pull-down menu.

Since the Flextext object treats \ as a special character, you must avoid using it except to express the embedded formatting codes described above. For example, displaying a CONFIG.SYS file in a Flextext object is not recommended because CONFIG.SYS files often contain the \ character. To see what we mean try:

```
{fyl(8, 1, "C:\config.sys")}
```

To solve this problem, you might consider replacing the \ character (ASCII 92) with another, such as a * (ASCII 42), via the Rpl() function, as in:

```
{rpl(fyl(8, 1, "C:\config.sys"), 92, 42)}
```

Also see Caption, FlexText Object

Textbox Object

Description The Textbox object displays one or more lines of text to the user.

Attributes
Alignment
AllowSelection
AnimPath
BackColor
BorderStyle
Bottom
Comment
Condition
Create
Destroy
DragMode
DrawPause
DrawShadow
DrawText
Enabled
FontBold
FontItalic
FontName
FontSize
FontStrikeThru
FontUnderline
ForeColor
Height
IDNumber
Indent
Initially
Left
MouseLeaveEvent
MouseOverEvent
MousePointer
Move
MultiLine
Right
SaveAsObject
ScrollBars
SelLength
SelStart
SelText
TabOrder
TabStop
Top
Update
Visible
Width
Zev
ZOrder

Details

Use a Textbox when you have several lines of text to display in a single font and foreground/background color combination. Contrast this with the more complex Flextext object that allows multiple colors and two fonts.

If the text does not fit inside the box, the user can scroll it up/down left/right (with or without ScrollBars). A Textbox can contain up to 32,000 characters.

Textbox text normally plots in a destructive fashion (i.e. it erases whatever it plots on top of). To display non-destructive (sometimes called transparent) text, use the A-pex3 PRINT and FONT commands, or see the DrawText attribute.

If you need a Textbox that is editable by the user, employ an Input Object.

There is a known bug in the Textbox object that can cause the cursor to appear in the wrong place when a vector font is used.

Also see

FONT, Input Object, PRINT

TextLength Attribute

Applies to	Input object
Description	Sets the maximum number of characters a user can enter in the editing area.
Settings	1 to 32,767
Also see	InputTemplate

THEN Command

Applies to	A-pex3 programming
Description	Marks the end of a conditional expression in an IF command.
Details	Refer to the IF command.

Tile Attribute

Applies to	Layout object
Description	Determines whether the BgndPicture image is replicated to fill the window. Available at design time only.
Settings	Yes replicate BgndPicture image No do not replicate BgndPicture image
Details	The Tile feature is often used with small BgndPictures to fill the background of the window with an attractive pattern. When Tile is enabled, the image is drawn with AutoRedraw temporarily enabled.
Notes	Tiled backgrounds can use a sizable amount of memory; error 480 can result if insufficient memory is available. Due to memory requirements, do not enable Tile for Scrollable windows.
Also see	BgndPicture, Wallpaper

Tim() Function

Applies to A-pex3 programming

Description Returns the current time, or a portion thereof.

Syntax tim(Which)

Details

When Which is	Tim() returns
""	HH:MM:SS
0	number of seconds past midnight
1	current hour (0 to 23)
2	current minute (0 to 59)
3	current second (0 to 59)
4	tick count mod 256 (approx. 18.2 ticks/second)

USER-DEFINED TIME STYLES

Employ the following characters in Which to create your own time formats. Be sure to surround the characters with quotes.

AM/PM	12-hour form with AM or PM
am/pm	12-hour form with am or pm
h	hour as a number without leading zero (0 to 23)
hh	hour as a number with leading zero (00 to 23)
n	minute as a number without leading zero (0 to 59)
nn	minute as a number with leading zero (00 to 59)
s	second as a number without leading zero (0 to 59)
ss	second as a number with leading zero (00 to 59)

Examples The following A-pex3 programming example displays the current time in HH:MM:SS format in the Textbox with IDNumber 1:

```
Textbox(1).Text = tim("")
```

The following example displays the current time in (H)H:MM am/pm format in the Textbox with IDNumber 1:

```
Textbox(1).Text = tim("h:nn am/pm")
```

Also see Dat() Function, Fmt() Function, PAUSE

TimeEvent Attribute

Applies to	Timer object
Description	Event code to generate, or programming to perform, each Period seconds.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	You can detect the event, and take other actions, via a Wait object.
Notes	Everest does not register TimeEvents while branching between pages (i.e. while not at a Wait object).
Also see	DrawPause, Period

TimeFormat Attribute

Applies to	Media object	
Description	Selects the format of position information in the StartAt and EndAt attributes.	
Settings	0	milliseconds
	1	hours, minutes, seconds (4 packed bytes)
	2	minutes, seconds, frames, unused (4 packed bytes)
	3	frames
	4	24-frame SMPTE
	5	25-frame SMPTE
	6	30-frame SMPTE
	7	30-drop-frame SMPTE
	8	bytes
	9	samples
	10	tracks, minutes, seconds, frame (4 packed bytes)
Details	Not all TimeFormats are available for all DeviceTypes. Choose the DeviceType before setting TimeFormat.	
Also see	DeviceType, EndAt, StartAt	

Timer Object

Description The Timer object can generate an event at a specified time interval.

Attributes
Comment
Condition
IDNumber
Initially
Name
Period
TimeEvent

Details Authors often use the Timer object for two purposes: 1) to set a time limit on a question, and 2) to periodically generate an event that can be used to perform an action (such as checking the location of the mouse in the window).

Also see PAUSE, Wait Object

TitleBar Attribute

Applies to	Layout object
Description	Determines whether a box with room for caption and various buttons is displayed at the top of a window.
Settings	Yes display title bar at top of window No do not display title bar
Details	The TitleBar and related elements are available only when WindowBorder has a value greater than 0.
Notes	Due to a bug in Windows, at project run time, a page that changes the state of the TitleBar of a window may precipitate errors if the window contains any objects (usually Button objects, and usually if WindowState is also changed). To work around this problem, place an Erase object (to remove all objects from the window) immediately before a Layout object that changes the TitleBar status. Change TitleBar at run time at your own risk. Due to another bug in Windows, when TitleBar is set to No, the Menu object's pull-down menu items do not plot correctly.
Also see	Caption, ControlBox, MaxButton, MinButton, SystemModal, WindowBorder

Top Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects

Description — Controls the vertical display location of the object.

Details — Specify in units of pixels. By default, the top edge of the window is 0.

The easiest way to adjust the Top attribute of an object at design time is via the VisualPage editor. First, click on the object to focus on it, then point to one of the sizing handles and either 1) press and hold down the left mouse button, and drag the mouse to move an edge, or 2) press and hold down the right mouse button, and drag the mouse to move the whole object.

For windows, the Top attribute is controlled via the Layout object. It specifies the distance from the top edge of the display.

Example — The following A-pex3 program centers the current window (window number 0) on the screen.

```
dwidth = gsm(0)          $$ width of display
wwidth = Window(0).Width$$ width of window
Window(0).Left = (dwidth - wwidth) \ 2
```

Also see — AutoCenter, Bottom, Height, Left, Move, Width

TopIndex Attribute

Applies to	Listbox object
Description	Determines which Item is displayed at the top of the list.
Details	The first item in the list is number 0.
Also see	ItemIndex, LookAt

TpColor Attribute

Applies to Picture object

Description Determines the color of the PictureFile image that is transparent. The transparency takes effect only at run time.

Double click Lets you click within the Picture object to visually select the transparent color. Before using this feature, set PictureFile to the name of the desired image file.

Details Wherever the transparent color appears in subsequently loaded PictureFile images, the background (i.e. the existing contents of the Picture object) will show through. When coupled with the CopyBgnd attribute, TpColor is the key to making irregularly shaped images appear superimposed over a BgndPicture.

Any color can be used as the transparent color, however, the most common settings for TpColor are:

&HFFFFFF&	makes white the transparent color
&H0&	makes black the transparent color

If you do not want any color of the PictureFile image to be transparent, leave TpColor empty.

Notes To see the effect of TpColor, run a preview of your page.

If the transparency fails for a .WMF type PictureFile, retry after setting the BackColor to the same value as TpColor.

The TpColor feature performs its magic with help from the Windows BitBlt function, and is therefore subject to that function's limitations. The most likely causes of GPF errors that might occur from the use of TpColor are bugs in the display driver (try a different display resolution and/or color depth in Windows) or insufficient memory.

TpColor temporarily enables AutoRedraw for the Picture object.

If the PictureFile attribute is empty, also leave TpColor empty (otherwise Everest might display an image left over from a prior page).

Also see CopyBgnd, CopyPic

Tries Attribute

Applies to	Button, Check, Combo, HScroll, Input, Listbox, Mask, Option, VScroll objects	
Description	Sets the number of chances a user has to respond correctly to the question, and enables automatic scoring.	
Settings	1 to 8	that number of tries
	9	retry until correct
	0	no tries (simply display)
	10	always retry (even after correct)
	empty	same as 10, but do not score in Sysvar(5) and Sysvar(6)

Details

The Tries attribute is useful in a CBT situation in which you want to judge the user's response for accuracy, and limit the number of times the user can attempt to answer correctly.

If you are not judging the user's response (via the Judge object) you should leave the Tries attribute empty. Tries is employed only when a Judge object judges the user's responses.

When Tries is set to a value of 0 through 9, and the number of Tries is exhausted or the user's response is judged as correct, and DisableObjs is enabled, Everest disables the object, preventing further user interaction with that object. If necessary, you can enable the object again by changing its Enabled attribute.

You can examine the value in Tries to provide custom feedback, or to determine when the user has answered correctly. Upon a correct response, Everest sets Tries to -1.

Notes

If you want to employ Everest's automatic scoring features, you must not leave the Tries attribute empty.

Also see

DisableObjs, Enabled, Judge Object

Typ() Function

Applies to A-pex3 programming

Description Returns a number that indicates the storage format Everest is using to hold the contents of the specified variable.

Syntax typ(VarName)

Details Everest stores the contents of variables in different ways depending upon the data. Strings of characters are stored as strings, but numeric values can be stored either as numbers or as strings of digits.

This table shows the numeric codes that the Typ() function can return:

CODE	INDICATES
0	Empty (variable not yet used)
2	Numeric integer
4	Single precision numeric
5	Double precision numeric
8	Character string
-1	Unknown

Examples The following example puts the value 8 in the variable named vartype because the variable named x contains a character string:

```
x = "123.5"
vartype = typ(x)
```

The following example puts the value 5 in the variable named vartype because Everest stores non-integer values in double-precision form:

```
x = 123.5
vartype = typ(x)
```

Also see Val() Function

Update Attribute

Applies to — Animate, Button, Check, Combo, Flextext, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, OLE, Option, Picture, SPicture, Textbox, VScroll objects

Description — Tells Everest to refresh the object's appearance on the page. Write-only. Available at run time only.

Details — To increase performance, Windows does not always immediately refresh an object's appearance on the page when you change an attribute. Instead, it sometimes waits for a "break in the action" to do so.

You can force Windows to refresh the object's appearance via the Update attribute. Simply set it to a value of 1.

Refer to the Initially attribute for a way to force a refresh of an object at design time.

Example — The following example displays the current time inside a Textbox with IDNumber 1. The use of Update assures that the display will be refreshed to show the current time:

```
DO
  Textbox(1).Text = tim("")
  Textbox(1).Update = 1        $$ forces refresh
LOOP IF ext(5) = 0             $$ while no event
```

Also see — Ext(101) Function, Initially, Visible

UpdateEvent Attribute

Applies to	Media object
Description	Event code to generate, or programming to perform, after each UpdateInterval time period elapses.
Settings	-32000 to 32000, or a string surrounded by quotes or any A-pex3 command except BRANCH, CALL, JUMP, OPEN and RETURN
Double click	Opens the Generate Keypress Event Code window. Press a key to generate the corresponding numeric event code.
Details	Authors often employ the UpdateEvent attribute to display the current status (such as Position) of the media DeviceType. UpdateEvent can also be used to coordinate multiple multimedia elements. For example, if you want to start another element, perhaps an animation, after the current Media element has played for 1 second, you would set UpdateInterval to 1000, and UpdateEvent to the action to perform, or event code to generate.
Example	The following example updates the Textbox with IDNumber 1 with the current position: `Textbox(1).Text = Media(1).Position`
Notes	If Wait is enabled, the UpdateEvent does not fire while the Command is being processed.
Also see	DoneEvent, UpdateInterval

UpdateInterval Attribute

Applies to	Media object
Description	Determines the length of time (in millseconds) between UpdateEvent events.
Settings	0 to 32000
Example	To fire an UpdateEvent every half second, set UpdateInterval to 500.
Notes	Due to a quirk in Windows, setting UpdateInterval to 0 may cause the Media object's buttons to not refresh properly.
Also see	UpdateEvent

Upr() Function

Applies to A-pex3 programming

Description Returns the character string String with characters a to z converted to the corresponding lower-case letters A to Z. Or, returns Numeric rounded to the nearest integer.

Syntax upr(String)
upr(Numeric)

Details Note that the Upr() function performs either one of two actions based on whether the parameter you pass is a character string or a number.

Example The following A-pex3 example puts initial capital letters at the start of the user's name:

```
uname = upr(sysvar(131) $\ 1) + (sysvar(131) $- 2) + " "
uname = uname + upr(sysvar(132) $\ 1) + (sysvar(132) $- 2)
```

Also see Fmt() Function, Lwr() Function, Val() Function

Val() Function

Applies to	A-pex3 programming
Description	Converts the leading numeric portion of a string to a number and returns that number.
Syntax	val(String)
Details	Since Everest automatically converts between strings and numbers based on context, rarely should you need the Val() function. Val() also has the ability to convert hexadecimal numbers to base 10 numbers. Prefix the hexadecimal number with &H.

Examples	The following calculation stores the number 123 in the variable named houseno:
	
	`houseno = val("123 Main Street")`
	
	The following calculation converts a hexadecimal number into a decimal one:
	
	`base10 = val("&HFF")`

Also see	Hex() Function, Operators, Typ() Function

Value Attribute

Applies to Button, Check, HScroll, Option, VScroll objects

Description Sets or returns the value represented by the object.

Settings **BUTTON and OPTION OBJECTS**

0 not selected
-1 selected

CHECK OBJECT

0 unchecked
1 checked
2 grayed

HSCROLL and VSCROLL OBJECTS

Anything between Min and Max.

Details For HScroll and VScroll objects, the Value you set in the Attributes window becomes the initial value of the bar. Use A-pex3 programming code to alter the Value at run time.

Example To increase the value displayed by a horizontal scroll bar with IDNumber 1 by 5 units, use the following in a Program object:

```
HScroll(1).Value = HScroll(1).Value + 5
```

Notes Changing the Value of an object can trigger an event (such as a ClickEvent for a Button). Use the State attribute if you wish to avoid this.

Also see Max, Min, State

Var() Function

Applies to	A-pex3 programming
Description	Indicates whether a variable exists.
Syntax	var("varname")
Details	The Var() function returns 0 if the variable does not exist, another number if it does. Be sure to surround the name of the variable with quotes. To create a new variable, simply refer to it in an A-pex3 programming command.
Example	The following A-pex3 example branches to the @start page in the LESSON27.ESL book if the variable by the same name does not exist: `IF var("lesson27") = 0 THEN BRANCH lesson27;@start`
Also see	Arr() Function, DELVAR, Typ() Function

Variables Window

The Variables Window is accessible via the main Author window's Window pull-down menu, as well as via the Debug Window. The Variables window lets you view, and edit, variables used by your project, and is especially helpful while debugging a running project. The cause of many programming problems is revealed by examining the values stored in variables.

VIEWING A VARIABLE

To view the contents of a variable, click on its name in the combo box. Everest will display the current value of the variable in the contents box. If the value is not what you expected, it may indicate a mistake in your A-pex3 programming. You might find the Debug window helpful for determining where a variable obtains its value.

FINDING CONTENTS

If you want to determine which variable contains a particular string of text or a number, click on the Find Contents button and follow the instructions. This feature can be handy while debugging your projects to help you determine the source of variable information.

CHANGING A VARIABLE

To change the value of a variable, first view it (as described above), then click in the contents box and modify it as desired. Closing the window or viewing another variable saves any changes you make.

SYSTEM VARIABLES

See the topic on System variables for more details about them.

RESET ALL

The reset all button removes all variables currently in memory. Authors use this feature to prevent old variable values from interfering with the project the next time it is test run in AUTHOR. It can also be handy after you edit an A-pex3 Program and change the number of array elements allocated in a DIM command. If you'd like to reset all variables at run time from within your project, refer to the DELVAR command.

Verb Attribute

Applies to	OLE object
Description	Specifies the operation to perform when an object is activated via Action 7.
Details	Use a number to specify the Verb. After setting the other attributes of the OLE object, click in the Attributes window to obtain more information, and Everest will display a list of the available verbs.
Also see	Action

VerticalAlignment Attribute

Applies to	Frame object	
Description	Controls the vertical position of the Caption in a Frame object.	
Settings	0	top
	1	bottom
	2	center
Notes	For VerticalAlignment to have effect, MultiLine must be set to No.	
Also see	Alignment	

VirtualHeight Attribute

Applies to	Layout object
Description	Determines the maximum vertical scrollable area of a window.
Settings	1 to 32000
Details	When VirtualHeight is greater than the height inside the window, and Scrollable is enabled, a vertical scroll bar appears in the window at run time. Note that the Height attribute includes the window border elements (such as the title bar), whereas VirtualHeight considers only the inside (the client area) of the window.
Also see	Scrollable, VirtualWidth, VValue

VirtualWidth Attribute

Applies to	Layout object
Description	Determines the maximum horizontal scrollable area of a window.
Settings	1 to 32000
Details	When VirtualWidth is greater than the width inside the window, and Scrollable is enabled, a horizontal scroll bar appears in the window at run time. Note that the Width attribute includes the window border elements, whereas VirtualWidth considers only the inside (the client area) of the window.
Also see	HValue, Scrollable, VirtualHeight

Visible Attribute

Applies to — Animate, Button, Check, Combo, Frame, Gauge, HScroll, Input, Layout, Line, Listbox, Mask, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects

Description — Determines whether an object is displayed.

Settings
- Yes — object is displayed
- No — object is hidden

Details — The Visible attribute is available at run time only. This means it is not listed in the Attributes window, and is accessible only via A-pex3 programming. Refer to the Initially attribute for a way to set Visible at design time.

The Visible attribute controls whether the object is being displayed somewhere. Even if an object is located off an edge of the window, it can still be considered visible because it would be displayed if simply relocated.

Visible provides a quick and easy way to hide an object, and make it visible later. While an object is hidden, you can reference all its attributes normally.

Also see — Enabled, Initially, SetFocus, Zev, ZOrder

VScroll Object

Description The VScroll object is a vertical slider bar with pointer that the user can adjust.

Attributes
Answers1
Answers2
AntIncorrect1
Bottom
ChangeEvent
CMIData
Comment
Condition
Create
Destroy
DragMode
Enabled
GotFocusEvent
Height
IDNumber
Ignore
Initially
JudgeVar
Judgment
LargeStep
Left
LostFocusEvent
Max
Min
MousePointer
Move
Name
Preset
ResponseVar
Right
SaveAsObject
SetFocus
Step
TabOrder
TabStop
Top
Tries
Update
Value
Visible
Width
Zev
ZOrder

Details Many authors employ an VScroll object to allow the user to choose from a range of numeric values. You can assign the value represented by the top and bottom edge of the bar.

Also see HScroll Object

VValue Attribute

Applies to	Layout object
Description	Determines the position of the pointer on the window's vertical scroll bar.
Settings	0 to VirtualHeight
Also see	HValue, Scrollable, VirtualHeight

Wait Attribute

Applies to	Media object
Description	Determines if Everest pauses until the Command finishes executing.
Settings	Yes wait until Command finishes No continue processing page
Example	Many authors use the Wait attribute to pause the project until a Play command has run to completion.
Also see	Command, UpdateEvent

Wait Object

Description Place a Wait object in the page to pause for user input.

Attributes
AllOtherAction
BackAction
BackActivator
Comment
CommentAction
CommentActivator
Condition
EventVar
HelpAction
HelpActivator
JudgeActivator
MenuAction
MenuActivator
Name
NextActivator
NextAction
Other1Action...Other8Action
Other1Activator...Other8Activator
Pause
QuitAction
QuitActivator
SaveAsObject

Details You must include a Wait object after interactive objects (such as Input and Check) in the page to give the user a chance to respond.

Even if the page does not contain any interactive objects, most authors employ a Wait object to wait for user input before continuing to another page.

Via the Wait object's various Activator attributes, you tell Everest what user actions (event codes) to trap. Via the corresponding Actions, you describe what operation to perform when an action is trapped.

Also see Pause

Wallpaper Attribute

Applies to	Button, Check, Frame, Gauge, Option objects
Description	Controls how a picture is displayed on the object.
Settings	0 scale image to fit object 1 display image in original size 2 replicate image to fill the object
Notes	Due to a bug in MicroHelp's Frame control, images are not scaled on Frame objects.
Also see	Pic, Tile

Width Attribute

Applies to	Animate, Button, Check, Combo, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects
Description	Controls the display width of the object.
Details	Specify in units of pixels. The easiest way to adjust the Width attribute of an object at design time is via the VisualPage editor. First, click on the object to focus on it, then point to one of the sizing handles, press and hold down the left mouse button, and drag the mouse. For windows, the Width is controlled via the Layout object. The Width includes the window border.
Also see	Height, Left, Move, Top

WindowBorder Attribute

Applies to	Layout object
Description	Controls the appearance of the edge of the window.
Settings	0 no border or related elements 1 single line, not user-sizable 2 double line, user-sizable 3 double line, not user-sizable
Notes	Changing the value of WindowBorder frequently from one page to the next is not recommended. When you change WindowBorder, Windows must recreate the window, causing flicker.
Also see	SystemModal, TitleBar

WindowLayer Attribute

Applies to Layout object

Description Controls whether the window appears on top of or beneath other windows.

Settings

0	Automatic (no change)
1	On top
2	Always on top
3	Top of ZOrder
4	Bottom of ZOrder

Details WindowLayer operates relative to all other open windows (including, for example, the Program Manager). If you only wish to change the top/bottom order of the window relative to other Everest windows, consider using ZOrder instead.

Also see Zev, ZOrder

WindowState Attribute

Applies to Layout object

Description Describes the visual state of a window.

Settings

-1	no change
0	normal
1	minimized (to an icon)
2	maximized (to full screen)

Details When you change WindowState, Windows must recreate the window; this produces some flicker. Avoid changing WindowState frequently.

The setting of WindowState in the Layout object is ignored if the window is already on the screen and Relocate is set to No. If necessary, you can force WindowState via A-pex3 programming.

Example The following A-pex3 commands make window number 2 normal size if it has been minimized:

```
IF Window(2)!Layout(1).WindowState = 1 THEN
  Window(2)!Layout(1).WindowState = 0
ENDIF
```

Also see Icon, Relocate, Scrollable, TitleBar

WindowStyle Attribute

Applies to	Layout object
Description	Returns or sets the window style attribute returned by the API GetWindowLong and SetWindowLong functions. Available at run time only. Intended for experienced programmers only.
Settings	Refer to a Windows API function Programmers Reference.
Details	This can be used to change several appearance attributes of a window, such as whether a TitleBar is present. Use at your own risk. Setting WindowStyle to improper values can cause Windows to operate erratically.
Also see	Hex() Function, Val() Function

WordWrap Attribute

Applies to	Input object	
Description	Determines whether text entered in the editing area is wrapped to the next line when it becomes too long to fit on the current line.	
Settings	Yes	word wrap text
	No	keep text on one line and scroll horizontally
Also see	Alignment	

Wrp() Function

Applies to A-pex3 programming

Description Returns a copy of the String parameter with Carriage Return/Line Feed (CR/LF) sequences inserted to break the text into lines of a desired length. Handy for printing purposes.

Syntax wrp(Length, String)

Details This function is particularly useful when you want to print the text of a Textbox object on the printer. For the Length parameter, specify a number greater than, less than or equal to 0, according to the following table:

Length	Wrp() Action
> 0	Places up to Length number of characters on a line. Inserts a CR/LF at the nearest preceding space. Good for non-proportional fonts, such as Courier.
< 0	Places up to abs(Length) text on a line, measured in current printer units. Inserts of CR/LF at the nearest preceding space. The default measurement unit is twips; for example, in Courier size 12 font, there are 144 twips per character. Good for proprotional fonts because it allows for varying character widths.
0	Returns String unchanged.

Example The following A-pex3 example prints the text of Textbox(1) on the printer, and wraps the text after 1000 twips per line:

```
LPRINT (2, wrp(-1000, Textbox(1).Text))
```

Also see LPRINTcommand

X1 Attribute

Applies to	Line object
Description	Determines the distance between the left edge of the window and one endpoint of the Line.
Settings	-32000 to 32000
Details	By default, the left edge of the window is 0.
Also see	X2

X2 Attribute

Applies to	Line object
Description	Determines the distance between the left edge of the window and one endpoint of the Line.
Settings	-32000 to 32000
Details	By default, the left edge of the window is 0.
Also see	X1

Y1 Attribute

Applies to	Line object
Description	Determines the distance between the top edge of the window and one endpoint of the Line.
Settings	-32000 to 32000
Details	By default, the top edge of the window is 0.
Also see	Y2

Y2 Attribute

Applies to	Line object
Description	Determines the distance between the top edge of the window and one endpoint of the Line.
Details	By default, the top edge of the window is 0.
Settings	-32000 to 32000
Also see	Y1

Zev Attribute

Applies to Button, Check, Combo, Gauge, HScroll, Input, Listbox, Mask, Media, Option, Picture, SPicture, Textbox, VScroll objects

Description Sets the ZOrder, Enabled and Visible attributes in one step. Write-only.

Settings Assign Zev a value of 0 to 7, as indicated in the table below:

	Visible	Enabled	ZOrder
0	N	N	top
1	Y	N	top
2	N	Y	top
3	Y	Y	top
4	N	N	bottom
5	Y	N	bottom
6	N	Y	bottom
7	Y	Y	bottom

Also see Enabled, Visible, ZOrder

Zip Maker Window

The Zip Maker Window is accessible via the main Author window's Utilities pull-down menu. The Zip Maker helps you copy files into compressed, .ZIP format.

FROM (SOURCE)

In the source section, highlight the files to compress. Tip: to quickly highlight all the files listed, double click on any one of them.

TO (DESTINATION)

In the destination section, designate where you want Everest to store the compressed version of the files.

ZIP Together Into One File - Enabling this tells Everest to combine all source files into one .ZIP file. You must enter the name of the destination .ZIP file in the field above this option

ZIP Each File Individually - Enabling this tells Everest create a different .ZIP file for each source file. So, for example, A.TXT and B.BMP will be compressed into A.ZIP and B.ZIP, respectively.

OTHER NOTES

If you make your book granular via theProject Packager, it is more efficient to enable the .ZIP option there than to not enable it there, and subsequently use the .ZIP maker to covert it to .ZIP form.

To create .ZIP files at run time, employ Fyl() function operation -7.

ZOrder Attribute

Applies to — Button, Check, Combo, Frame, Gauge, HScroll, Input, Layout, Listbox, Mask, Media, OLE, Option, Picture, Shape, SPicture, Textbox, VScroll objects

Description — Forces a visible object to the front or back of its layer. Write-only at run time.

Settings

-1	no change
0	bring object to front
1	push object to back

Details — Think of the objects on the page as a stack of pancakes viewed from the top. When two pancakes are exactly on top of each other, only the one on top is visible. Sometimes two pancakes overlap, in which case part of one can be seen beneath the other.

The ZOrder attribute lets you rearrange the pancakes in the stack. You can either bring a pancake to the top of the stack (setting 0), or move it to the bottom (setting 1).

The ZOrder attribute defaults to 0 (bring to front). Unless you change this value, it means Everest will put each object it encounters on the top of the "pancake stack."

In special situations, you might want to manually move an object to the front (so that it is not obscured by other objects), or to the back (to hide it behind other objects). This can be done by setting ZOrder via A-pex3 programming.

It is important to note that ZOrder works only within a particular layer. Here are the layers:

Deepest: image loaded via BgndPicture attribute of the Layout object

Mid-Deep: A-pex3 Xgraphics commands in window

Mid-Top: Shape and Line objects

Topmost: all other objects with a visible component

Example — The following example moves the Textbox with IDNumber 1 to the front:

```
Textbox(1).ZOrder = 0
```

Also see — Initially, SetFocus, Update, Visible, WindowLayer, Zev

Appendix A - Key Press Event & ASCII Codes

The following chart shows the event codes that are generated by various key presses and mouse clicks. Everest mouse codes in Sysvar(11), and key press codes in Sysvar(12). (For the proper codes to use with the Chr() and Asc() functions, refer to the ASCII codes table found later in this topic.)

Left mouse	1 (button click)
Right mouse	2 (button click)
Middle mouse	4 (button click)
Backspace	8
Tab	9
Center 5	12
Enter	13
Shift alone	16
Ctrl alone	17
Alt alone	18
Pause	19
CapsLock	20
Escape	27
Space	32
PgUp	33
PgDn	34
End	35
Home	36
Left	37
Up	38
Right	39
Down	40
PrtSc	44
Ins	45
Del	46
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
a	65
b	66
c	67
d	68
e	69
f	70
g	71
h	72
i	73
j	74

k	75
l	76
m	77
n	78
o	79
p	80
q	81
r	82
s	83
t	84
u	85
v	86
w	87
x	88
y	89
z	90
numeric 0	96
numeric 1	97
numeric 2	98
numeric 3	99
numeric 4	100
numeric 5	101
numeric 6	102
numeric 7	103
numeric 8	104
numeric 9	105
numeric *	106
numeric +	107
numeric -	109
Decimal	110
numeric /	111
F1	112
F2	113
F3	114
F4	115
F5	116
F6	117
F7	118
F8	119
F9	120
F10	121
F11	122
F12	123
NumLock	144
ScrollLock	145
;	186
=	187
,	188
-	189
.	190
/	191

`	192
[	219
\	220
]	221
'	222
Shift	add 1000
Ctrl	add 2000
Alt	add 4000
Key Up	add 8000 (enable by setting Sysvar(170) = 8)

Examples:

Key	Code Generated
a	65
Shift+A	1065
Ctrl+A	2065
Alt+A	4065
Ctrl+Alt+A	6065

Use the key(5, EventCode) function to convert an event code to its ASCII equivalent. For example, key(5, 65) returns 97 (the ASCII code of the lower-case letter a). If an ASCII equivalent does not exist, the Key() Function negates and returns the EventCode parameter.

Error! Reference source not found. When editing any of the xxxEvent or xxxActivator attributes, you can have Everest automatically generate the event code for a particular key. To do so, double click on the attribute name in the Attributes window; when the "Generate Keypress Event Code" window appears, press the desired key.

ASCII CODES

The following table shows the codes used by the Chr() and Asc() functions:

Tab	9
Line Feed	10
Enter	13 (also known as Carriage Return)
Escape	27
Space	32
!	33
"	34
#	35
$	36
%	37
&	38
'	39
(	40
)	41
*	42
+	43
,	44

-	45
.	46
/	47
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
[	91
\	92
]	93
^	94
_	95
`	96
a	97
b	98
c	99
d	100
e	101
f	102

g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122
{	123
|	124
}	125
~	126
Others	127 to 255

Appendix B - Bounds

Windows/Pages

Max windows open simultaneously at run time	8
Max memory for objects per window	64000 bytes
Max page size	32000 bytes
Max pages per book	4000

Attributes

Max length of Text attribute	32000 chars
Max length of other string attributes	250 chars

Variables

Max unique names for author defined variables	2500
Max number of variables and array elements	32000
Max memory for author defined variables	Windows memory
Max memory for author defined variables at log off	64000 bytes
Max memory for system variables	Windows memory
Max memory for system variables at log off	64000 bytes
Max string length	32000 chars (1 char = 1 byte)
Min numeric value	-3.4 x 10^38
Max numeric value	3.4 x 10^38

A-pex3 Program Code

Max program size	32000 chars
Max line length	250 chars
Max function and parentheses nesting	8 levels

External Snap-ons

Max snap-ons	128
Max snap-on driver programs	16
Max snap-ons per driver	8

Appendix C - Error Messages

GPF Errors

Global protection fault errors (better known as GPFs) are the result of bugs in Windows, other Microsoft software, or other third-party software. They can result from unexpected values in memory, as well as a host of other causes. Video display drivers are notoriously buggy and are often the cause of GPFs; you might try running Windows in a different resolution and/or color depth to see if that cures the GPF. Since GPFs are outside Everest, there's little we can do to correct them. If you can repeat a GPF at will, let us know how: we'll need to know step-by-step instructions for creating a page from scratch that repeats the problem. If we can also duplicate it, there's a small possibility that we can adjust Everest to work around it.

Everest Specific Errors (codes less than 0):

These are error messages generated by Everest to inform you of a problem in your project. If you are unable to resolve the problem, try creating a page from scratch. If you still cannot find the source of the trouble, call tech support; we'll need to know step-by-step instructions for creating a page from scratch that duplicates the problem.

-098 Renamed application
: Do not change the name of the Everest .EXE files.

-099 Contact tech support
: Report this error to technical support.

-112 Insufficient memory for operation
: There is not enough memory available to perform an action.

-114 File not found
: A file was not found on disk.

-119 Internal error
: A problem occurred in Everest. Please report this error to tech support.

-123 Page name missing
: Everest expected you to enter the name of a page.

-131 Trouble reading current directory
: Everest had trouble reading the files on disk. This could indicate a disk problem.

-134 Illegal or undefined variable name
: Variable names must start with a letter, and may contain letters, numbers and the underline character (ASCII 95).

-135 Illegal variable number
: You attempted to reference an array variable without specifying an element number.

-136 Variable not defined as array
: You attempted to use a non-array variable where only an array is allowed.

-137 Out of array bounds
: You attempted to reference an element outside the bounds of an array. For example, xyz(15) when the xyz array contains only 10 elements. Check your DIM and REDIM commands. Use the Variables window to determine the number of elements currently in the array. Remember that DIM does not change the number of elements in an array if that array already exists.

-138 Expression conclusion ("then") missing
: The "THEN" part of a conditional expression is missing.

-139 Illegal character
: In a page or object name, you attempted to use a character that is not allowed. Or, in an expression, Everest expected an operator such as = or +, but found some other character.

-142 Closing quote mark missing

Everest found an open quotation mark, but no closing one before the end of the expression. If you need to put a quotation mark in a string, use Chr(34).

-143 Range error

A value was outside an acceptable range. Example: char = "Everest"\-1 produces this error because you cannot parse the leftmost -1 characters of a string.

-144 = sign or space expected

Use = to assign a value to a variable. This error can also occur if you omit the IF command in a conditional expression, or omit the space between a command and its parameter(s).

-145 Bad function or mismatched ()

This indicates an expression does not have the same number of open and closing parentheses. It can also indicate that functions are nested more than 8 levels deep.

-146 Relational operator expected

Everest expected a relational operator in an expression, but did not find one.

-147 Constant or variable expected

An expression ended sooner than Everest expected.

-149 Unexpected character

Everest did not expect to find this character in this context. For example, using & in the non-conditional portion of an expression will produce this error.

-154 Mismatched parentheses

This indicates an expression does not have the same number of open and closing parentheses.

-158 Page not found

A page you attempted to load or branch to does not exist. Check the page name and current book. In the course of creating your project, this error is normal during test runs when you have not yet created the page to which the project is attempting to branch.

-160 Book is full

There is no more room in the book. Note that special versions of Everest (such as the working model and evaluation package) have an artificial maximum limit on the number of pages in a book. The actual version of Everest is limited only by memory and disk space in the computer.

-161 Not available on your hardware

This operation is not available on this computer (due to hardware limitations).

-162 Illegal math operation

Everest detected an illegal operator in an expression.

-163 IF...THEN block missing ENDIF

The ENDIF command is missing.

-164 IF...THEN block not allowed here

Block-style IF...THEN expressions are not allowed here.

-165 Page name too long

Page names are limited to 8 characters. Object names can contain 19 characters.

-167 System help file not found

Everest was not able to locate the EVEREST.HLP file.

-168 System help page not found

Everest could not locate a topic in the help file.

-173 Expression too lengthy to evaluate

A-pex3 expressions are limited to 250 characters in length.

-174 WARNING: Page name mismatch

Each time Everest loads a page from disk, it double checks the name of the page that was loaded. This message indicates the name of the page loaded from disk does not match the name Everest was expecting. This error usually indicates a damaged book (.ESL file) due to a disk error. You should probably work from a backup copy of the book, as other pages may also be damaged.

-181 This item may not be edited

This page or object may not be changed.

-185 Process terminated by user

An informational message that indicates you stopped a process before it ran to completion.

-199 Empty page slot

Everest attempted to retrieve a page, but the page or book either was missing or was empty. Usually, this error means the page you attempted to load or branch to was not found. Check the page name and current book. In the course of creating your project, this error is normal during test runs when you have not yet created the page to which the project is attempting to branch.

-201 Exceeded author defined var slots

There is no more room to hold author defined variables or array elements.

-202 Improperly initialized user record

This user's record was not properly initialized. Delete it via the INSTRUCT program, and recreate it.

-203 Duplicate user record

Two records in the user records database are identical. Please report this to tech support.

-204 No record in database

Everest expected to find a user's record in the database, but did not. Please report this to tech support.

-205 Source and destination must be different

You cannot perform this operation (such as copying a book) within a given disk subdirectory.

-208 Bad internal message number

Everest used an incorrect message number. Please report this to tech support.

-209 Entry required here

Everest was expecting an entry in a field, but found none.

-210 Item deleted at another station

Before an item could be retrieved, it was deleted at another station (on the network).

-211 Not available in this version of Everest

This feature is not available in this version of Everest. You may need a newer version.

-212 DOS 3.0 is required

Everest requires DOS 3.0 as a minimum.

-213 GOTO label not found

A LABEL for a GOTO command was not found. A LABEL with the name specified in the GOTO command must exist in the same Program object.

-214 Reserved word

The names of variables may not be the same as the names of A-pex3 commands or functions.

-215 Too many parameters for function

This function does not accept this many parameters.

-216 Nested DO...LOOP not allowed

Currently, DO...LOOPs may not be used inside other DO...LOOPs.

-217 (RE)LOOP without DO

A LOOP or RELOOP command was detected outside a DO...LOOP.

-218 No such operation

This action does not exist.

-219 Insufficient region coordinates

The Reg() function or =T= operator did not find the expected number of X-Y coordinates (typically a single or double pair of coordinates is required).

-220 Missing LOOP

A DO command did not find a corresponding LOOP command.

-221 Illegal char in var name

Everest found an illegal character in the name of a variable. Variable names must start with a letter, and may contain letters, numbers and the underline character (ASCII 95).

-222 Decimal point already used
In a numeric expression, Everest found more than one decimal point.
-223 Label not allowed inside IF or DO
The LABEL command is not allowed inside an IF...THEN block or DO...LOOP construct.
-224 Label must be at leftmost edge of line
Do not indent LABEL commands.
-225 Undefined variable
This variable has not yet been defined.
-226 Undefined attribute
This attribute is unknown.
-227 No such object name
This object class name is unknown
-228 No such attribute name
This attribute is unknown.
-229 Parameter list must be enclosed with ()
The list of parameters for A-pex3 commands must be enclosed with parentheses.
-230 Max 20 rows on pulldown menu
Pull-down menus are limited to 20 rows.
-231 Max 8 columns on pulldown menu
Pull-down menus are limited to 8 columns.
-232 Bad object number
An object with this number does not exist. Please report this to tech support.
-233 JLabel of JUMP not found
A JLabel object for the target of a JUMP was not found in this page. Use the Name attribute of the JLabel object in the JUMP command.
-234 HyperHelp Error
The Windows help system returned an error.
-235 Includes nested too deeply
Include objects can be nested up to 8 levels deep.
-236 Wait object not allowed in included page
A Wait object may not appear in a page that is used by an Include object.
-237 MCI error detected
The Windows Media Control Interface detected an error. Usually, additional error information is displayed. This error has several possible causes; it will occur, for example, if you try to play a Video for Windows .AVI file, but neglect to specify a FileName. If you cannot resolve the error, it may indicate trouble with the drivers for a multimedia device.
-238 Exceeded max # of variable names
The limit on the number of unique variable names has been reached.
-239 Exceeded max # objects in one class
The limit on the number of objects in a given class has been reached.
-240 Cut & paste buffer is full
There is no more room in the cut & paste buffer.
-241 Backup stack is empty (can't do @prev branch)
A BRANCH @prev command was encountered, but there is no page in the list of previous pages (the backup stack). Consider making the BRANCH command conditional, such as IF Len(sysvar(71)) > 0 THEN BRANCH @prev.
-242 Menu stack is empty (can't do @menu branch)
A BRANCH @menu command was encountered, but there is no page in the list of menu pages (the menu stack). Consider making the BRANCH command conditional, such as `IF Len(sysvar(81)) > 0 THEN BRANCH @menu.`
-243 Call stack is empty (can't do return)

A RETURN command was encountered, but no corresponding CALL command had been used previously.

-244 Book stack is corrupted
The list of books has been damaged.

-245 Incorrect create syntax
The syntax of the Create or Destroy attribute is incorrect.

-246 Bad ID# (must be from 1 to 99)
The IDNumber attribute must range from 1 to 99.

-247 Page is missing objects
Everest was not able to load all objects from disk.

-248 Can't find object in page
Everest could not find an object in the page.

-249 Unable to load page from disk
The page could not be loaded. This can indicate a damaged book.

-250 Feature no longer available
This feature is not available in this version of Everest.

-251 No such object class
The class name is bad.

-252 Unable to load object from disk
The object could not be loaded.

-253 .ESL has bad file format
The book file appears to be damaged. You may need to work from a back up copy.

-254 Too much data for linklist
The linked list database can handle records up to 32,000 bytes.

-255 Page too large to save
This page contains too much text to save in its current form. Change the SaveAsObject attribute of one or more objects (particularly large Program objects) to Yes, and retry.

-256 Window number must be from 1 to 8
Everest encountered a window number outside of the allowed range.

-257 CALL stack is corrupt
The list of pages on the CALL stack is damaged.

-258 Error during RTF file load
Everest was not able to load a Rich-Text Format file.

-259 No project running (for DDE request)
The Dde() function attempted a conversation with an application that is not currently running.

-260 Waiting for access to file on network. Continue to try for access?
Informational message. Everest is attempting to access a shared file that is currently locked by another user on the network. If you do not continue to try, error 70 will be generated.

-261 IF required for conditional DO or LOOP
A conditional expression in a DO or LOOP command must start with IF.

-262 Comment file name extension must be .ECM
The name of the comments file must end with .ECM

-263 Records file name extension must be .EUR
The name of the user records file must end with .EUR

-264 External reply timed out
Everest sent a message to an external object, but did not receive a timely reply.

-265 No external object with this name
You attempted to load an object for which there is no definition. External object classes must be defined via External add-ons. You can specify such add-ons by loading them via Add External in the File pull-down menu, or via "External=" entries in the EVEREST.INI file.

-266 Too many external files

Everest can load up to 16 External add-ons with 8 objects each at one time.

-267 Same external file loaded previously

Only one copy of an External add-on can be loaded in memory at a time.

-268 Book file name extension must be .ESL

The file name extension of Everest book files must be .ESL.

-269 Exceeded IF block nesting level

IF blocks can be nested up to 8 levels deep.

-270 Exceeded DO...LOOP nesting level

DO...LOOPs can be nested up to 8 levels deep.

-271 Block IF without ENDIF

IF blocks must end with an ENDIF command.

-272 DO without LOOP

DO...LOOP structures must end with a LOOP command.

-273 Not an Everest add-on module

The external .EXE you attempted to load does not contain the Everest add-on signature.

-274 File not found in book

An embedded file was not found in the book.

-275 Windows Help error

The Windows Help system reported an error. This could be caused by low memory or resources.

-277 String too long

Everest encountered a string that was too long to handle.

-278 User record number mismatch

The user record number did not agree with the copy read from the user records file. This could indicate a damaged user records file, or unauthorized tampering with the user records.

-279 Please highlight (select) something before using this feature

This feature requires that you first choose the item or text on which to act. Do so, then retry.

-280 Newer version needed

You will need to upgrade to a newer version of Everest in order to edit this page.

-281 Cannot be edited with this version

The book you are attempting to edit (load or save a page) may not be edited with the version of Everest you are using. If you are using the Everest Free Authoring Software, remember that it can only be used to edit your own books.

-282 Protection failure 0

Make sure the key is securely attached to the parallel port.

-283 Protection failure 1

Make sure the key is securely attached to the parallel port.

-284 Protection failure 2

Make sure the key is securely attached to the parallel port.

-285 Improper spacing

Blank spaces are rarely significant in A-pex3 programming, but you've found a place they are. This can be caused by a space between an object and attribute name. The proper syntax is <object>.<attribute> with no blank spaces in between.

-286 Page already exists

A page by this name already exists. Choose a different name.

-287 Expected attribute name

In A-pex3 programming, object references are followed by a period and an attribute name. This error can be caused by a blank space or other illegal character between the period and attribute name.

-288 List full; if possible, narrow search

The size of a list exceeded 64K. If possible, narrow the search parameters so that the list will be shorter.

-289 Unexpectedly unable to find object

An object that should have been present in the page was not found. Please report the conditions under which this occurs.

-290 Comment list full

There is no more room to store comments about pages.

-291 Error during .ZIP codec

An error occurred while either compressing or uncompressing a .ZIP file. Check the file with PkZip.

-292 No page is open

The operation you attempted to perform requires an open page. Double click on a page in the book editor, or create a new page.

-293 AnimPath is too long

AnimPath strings are limited to 1024 characters.

-294 A book or page by this name already exists

You tried to assign a name to a book or page, but that name is already in use. Choose another name.

-295 Cannot drag & drop; use cut & paste instead

The action you attempted to perform is not supported via drag and drop. Instead, highlight the desired objects and either cut or copy them, then paste them at the desired location.

-296 Cannot append from different books

The Append feature cannot be used on two different books. Use Copy instead.

-297 That operation not allowed on books

The operation you attempted to perform is not allowed on books.

-298 Open a book before performing this operation

The operation you attempted requires that a book be open. Double click on a book in the Book Editor to open it.

-299 Page and/or book must be closed first

The operation you attempted requires that the page and/or book be closed. Double click on the open page or book to close it.

-301 A-pex3 Compiler framing error

The compiler encountered an illegal character in the compiled code. This could indicate a problem with the compiler. Please report the line of programming that triggers this error.

-304 Command not allowed here

The A-pex3 command you are using is not allowed in this location.

-307 Expected attribute

The compiler expected an attribute name, but found none. This could happen if you enter the name of an object without an attribute. Please report the line of programming that triggers this error.

-308 Unexpected attribute

The compiler encountered an unexpected reference to an object's attribute. Please report the line of programming that triggers this error.

-309 Can't find object

Everest was unable to find an object you referenced in your programming. This error can be caused by an attempt to GOSUB to a Program object that does not have SaveAsObject enabled. Also, check that you have correctly spelled the name of the Program.

-310 Problem with object counter

Internal error. Please report how to duplicate this to tech support.

-311 Unable to open/find book

Everest was unable to locate the book to open. Please report how to duplicate this to tech support.

-312 Name must start with a letter

The name of this item must start with a letter (a to z).

-313 Page order invalid

The order of the pages in this book has been lost or damaged. Everest will attempt to recover as much as possible, placing "lost" pages at the end of the book. In rare cases, you may need to work from a backup copy of the book.

-314 Unable to branch to @next or @back

Everest was unable to locate the page to branch to for a BRANCH @next or BRANCH @back command. This could happen if, for example, you use BRANCH @next from the last page in a book.

-315 Socket exception error

An unrecoverable error occurred during Inter/intranet communication.

-316 Socket close error

An error occurred while closing an Inter/intranet socket connection.

-317 Socket timeout

Everest waited to hear a reply from the Inter/intranet site, but none arrived; the communication channel might be blocked or disconnected. To tell Everest to wait longer, set Sysvar(181) to the desired number of seconds to wait.

-318 Maximum sockets already in use

Too many concurrent Inter/intranet connections have been established. Please report to tech support the conditions under which this error occurs.

-319 Bad host name

The name of the Inter/intranet host is unacceptable.

-320 File not found on host

Everest attempted to download the file you wanted from the host, but no such file was found. Check the name, and verify the host has that file for downloading.

-321 Error during FTP upload to host

An error occurred during the attempt to upload a file to the host. Your FTP Settings may be incorrect.

-322 File to upload is missing or empty

Everest attempted to upload a file, but could not find it, or the file was empty (length 0).

-323 Upload timeout

Everest waited to hear a reply from the FTP upload server, but none arrived; the communication channel might be blocked or disconnected. To tell Everest to wait longer, set Sysvar(181) to the desired number of seconds to wait.

-324 Incompatible file

This file cannot be loaded because Everest cannot understand its contents. This error can occur, for example, if you attempt to display a graphics file stored in a format that is not supported.

-325 Download queue error

While a file is being downloaded, if a request comes in to download another, Everest will sometimes add it to a list for downloading momentarily. This error indicates something was wrong with the list.

-326 Incorrect book password

The password you entered is not the correct one.

-327 Bad link in file (incomplete data)

There is bad data in the .ESL or .EUR file. You may need to work from a backup. You can also try copying the contents of the file into a new file; for books, use the Copy Pages utility.

-328 Bad link in file (expected more data)

See error -327.

-329 Bad data in file (corrupted?)

See error -327.

-330 Cache and source location must be different

The location of the cache and the source cannot be identical. Use a different location for one or both.

-331 Unable to make directory

Everest could not create a subdirectory. Remember that Everest can only create subdirectories that are one level below existing directories.

-332 This book cannot be used with this version

Your version of Everest cannot use this book. For example, the Everest Free-Authoring Software cannot be used to modify or view books created with other versions. ERUN cannot be used with books created with the Everest Free-Authoring Software.

-333 Choose an item from the list

Before you can proceed, you must choose (highlight) one of the items in the list displayed.

-334 Bad link in file (bad first pointer)

There is bad data in the .ESL or .EUR file. You may need to work from a backup. You can also try copying the contents of the file into a new file; for books, use the Copy Pages utility.

-335 Cannot paste there

That location is not acceptable for pasting. For example, you cannot paste in the Book Editor above the first book listed; drag the pointer lower and retry.

-336 Please specify a page name

Everest was expecting you to enter the name of a page, but you either omitted it or left the page field empty.

Microsoft Error Messages (plus descriptions for the most commonly encountered ones):

001 NEXT without FOR
002 Syntax error
003 RETURN without GOSUB
004 Out of DATA
005 Illegal function call

This is Microsoft's "catch-all" error message for all errors that do not fit in another category. Please note how the error can be repeated from scratch, and report it to tech support.

006 Overflow
007 Out of memory

Usually this is caused by too many objects in a window. Another possible cause is insufficient available RAM in the computer.

008 Label not defined
009 Subscript out of range
010 Duplicate definition
011 Division by zero
012 Illegal in direct mode
013 Type mismatch
014 Out of string space

This typically results from too much data stored in system and author defined variables.

016 String formula too complex
017 Cannot continue
018 Function not defined
019 No RESUME
020 RESUME without error
021 Unprintable error
022 Missing operand
023 Line too long

024 Device timeout
025 Device fault
026 FOR without NEXT
027 Out of paper
028 Out of stack space
029 WHILE without WEND
030 WEND without WHILE
033 Duplicate LABEL
034 Disk Full
035 Subprogram not defined
036 Subprogram already in use
037 Argumentcount mismatch
038 Array not defined
039 CASE ELSE expected
040 Variable required
044 Disk is writeprotected
045 File is locked
046 Volume is locked
047 File is busy (delete)
048 Error in loading DLL
049 Bad DLL calling convention
051 Microsoft internal error
052 Bad filename or number
053 File not found
054 Bad file mode
055 File already open
056 FIELD statement active
057 Device I/O error

Usually the result of a disk drive error.

058 File already exists
059 Bad record length
061 Disk full
062 Input past end of file

Fyl() function operation 11 returns this error code after it has read the entire file. If you are looping and reading records from the file, this error is to be expected; simply exit the loop upon encountering this error.

063 Bad record number
064 Bad filename
067 Too many files
068 Device unavailable
069 Communicationbuffer overflow
070 Permission denied

This error code is returned when you attempt to open a file that is locked by another station on the network. Retry opening the file until it becomes available (i.e. until no error code is returned).

071 Disk not ready
072 Diskmedia error
073 Feature unavailable
074 Rename across disks
075 Path/File access error

This is often caused by using an illegal disk path in the EVEREST.INI file (for example, referring to a drive or subdirectory that does not exist). The SPicture object has been known to return this

error if the SPictureFile is missing.

076 Path not found
080 Feature removed
081 Invalid filename
082 Table not found
083 Index not found
084 Invalid column
085 No current record
086 Duplicate value for unique index
087 Invalid operation on null index
088 Database needs repair
091 Object variable not set
092 FOR loop not initialized
093 Invalid pattern string
094 Invalid use of Null
095 Cannot destroy active form instance
260 No timer available
280 DDE channel not fully closed
281 No more DDE channels
282 No foreign application responded to a DDE initiate
283 Multiple applications responded to a DDE initiate
284 DDE channel locked
285 Foreign application won't perform DDE method or operation
286 Timeout while waiting for DDE response
287 User pressed Esc during DDE operation
288 Destination is busy
289 Data not provided in DDE operation
290 Data in wrong format
291 Foreign application quit
292 DDE conversation closed or changed
293 DDE method invoked with no channel open
294 Invalid DDE Link format
295 Message queue filled: DDE message lost
296 PasteLink already performed on this object
297 Cannot set LinkMode; invalid LinkItem or LinkTopic
298 DDE requires DDEML.DLL
320 Cannot use character device names in filenames
321 Invalid file format
340 Control array element ## does not exist

This error can result from referring via A-pex3 programming to an object that does not (yet) exist in the window. Remember that at run time the page is executed in order from top to bottom. Be sure to position Program objects in the page AFTER the objects to which their A-pex3 programming refers. Also, check that you are employing the correct IDNumber.

341 Invalid object array index
342 Not enough room to allocate control array
343 Object not an array
344 Must specify index for object array
345 No more objects allowed in this window
360 Object already loaded

Unrepeatable errors of this type are due to a bug in Windows.

361 Cannot load or unload this object

362 Cannot unload controls created at design time
363 Custom control not found
364 Object was unloaded
365 Unable to unload within this context
366 No MDI form available to load
380 Invalid attribute value
381 Invalid attribute array index
382 Attribute cannot be set at run time
383 Attribute is read-only
384 Attribute cannot be modified when window minimized or maximized
385 Must specify index when using property array
386 Attribute not available at run time
387 Attribute cannot be set on this object
388 Cannot set Visible attribute from a parent menu
389 Invalid key
390 No defined value
391 Name not available
392 MDI child windows cannot be hidden
393 Attribute cannot be read at run time
394 Attribute is write-only
395 Cannot use separator bar as menu name
400 Window already displayed
401 Cannot show non-modal window when modal window is already displayed
402 Must close or hide topmost modal window first
403 MDI windows cannot be shown modally
404 MDI child windows cannot be shown modally
420 Invalid object reference
421 Method not applicable for this object
422 Attribute not found
423 Attribute or control not found
424 Object required
425 Invalid object user
426 Only one MDI window allowed
460 Invalid Clipboard format
461 Specified format does not match that of data
480 Cannot create AutoRedraw image

The AutoRedraw image is the picture that appears in the background of a window or a Picture object. If there is insufficient memory for this image, error 480 can result. To resolve the problem: make more memory available to Windows; disable AutoRedraw in the Layout object; avoid SpecialEffects; and/or disable Tile.

481 Invalid picture
482 Printer error
520 Cannot empty Clipboard
521 Cannot open Clipboard

Inter/intranet related

20000 Connection closed
20001 Memory allocation error
20002 Socket is already closed
20003 Socket is aready listening

20004 No port number or service name specified
20005 Socket is already connected
20006 UDP Socket is already active
20010 Winsock DLL not found
20011 Socket is not closed
21004 Interrupted system call
21009 Bad file number
21013 Permission denied
21014 Bad address
21022 Invalid argument
21024 Too many open files
21035 Operation would block
21036 Operation now in progress
21037 Operation already in progress
21038 Socket operation on non-socket
21039 Destination address required
21040 Message too long
21041 Protocol wrong type for socket
21042 Bad protocol option
21043 Protocol not supported
21044 Socket type not supported
21045 Operation not supported on socket
21046 Protocol family not supported
21047 Address family not supported by protocol family
21048 Address already in use
21049 Can't assign requested address
21050 Network is down
21051 Network is unreachable
21052 Net dropped connection or reset
21053 Software caused connection abort
21054 Connection reset by peer
21055 No buffer space available
21056 Socket is already connected
21057 Socket is not connected
21058 Can't send after socket shutdown
21059 Too many references, can't splice
21060 Connection timed out
21061 Connection refused
21062 Too many levels of symbolic links
21063 File name too long
21064 Host is down
21065 No Route to Host
21066 Directory not empty
21067 Too many processes
21068 Too many users
21069 Disc Quota Exceeded
21070 Stale NFS file handle
21071 Too many levels of remote in path
21091 Network SubSystem is unavailable
21092 WINSOCK DLL Version out of range
21093 Successful WSASTARTUP not yet performed

22001 Host not found
22002 Non-Authoritative Host not found
22003 Non-Recoverable errors: FORMERR, REFUSED, NOTIMP
22004 Valid name, no data record of requested type

Appendix D - Known Bugs and Quirks

The following is a list of known bugs and quirks in software external to Everest (i.e. in Windows, Visual Basic, .VBX and .DLL files). These bugs have been reported to the appropriate developer. If and when they update their software, we will update Everest as soon as is practicable.

An audible warning may sound when Ctrl+M is pressed. (Microsoft)

While authoring, clicking on a Media object in the VisualPage editor might not move the focus to the object. Instead, try clicking on the desired object in the Book Editor. (Microsoft and MicroHelp)

An Out of String Space error might be generated when adequate memory is indeed available. (Microsoft)

The focus can move from one object to another when you choose a color from the color dialog box. (Microsoft)

The ClickEvent for an object may not fire if that object is located on top of a Flextext object. (Crescent)

Very sluggish performance or malfunctioning Jump and Popup words of a Flextext object might be caused by a Flextext object that is stuck in an inifinite replot loop (you might be able to observe its scroll bar flickering). If this occurs, try adding an empty line of text to the offending Flextext object. (Crescent)

The Tab key might not move the focus to the appropriate object. (Microsoft)

For Input and Textbox objects, when BorderStyle is 2 and scroll bars are displayed, the window's background color might show within a portion of the object. (MicroHelp)

Due to a bug in Windows, avoid using function Ext(101) while processing a ChangeEvent. (Microsoft)

Due to a bug in Windows, avoid using the Ext(101), Ibx() or Mbx() functions in a Program object that has Refresh enabled. (Microsoft)

Due to a limitation in Windows, avoid using the Ibx() and Mbx() functions while processing a ChangeEvent. (Microsoft)

Due to a bug in Windows, the DoneEvent might not be generated if the Mbx() or Ibx() functions are in use at the time of the event. (Microsoft)

Due to a bug in Windows, TabStop must be enabled for HScroll and VScroll objects to receive the focus via a mouse click. (Microsoft)

Due to a bug in Windows, do not have the Debug window open while processing a ChangeEvent. (Microsoft)

For the Animate object, enabling FadeIn or FadeOut can cause the animation image to become invisible unexpectedly. (Autodesk)

The AnimStoppedEvent for an Animate object may not fire if the object is located off an edge of the

window; note that setting Initially to 0 causes the Animate object to be moved off the edge of the window (due to other Animate object quirks). (Autodesk)

The cursor may not appear in the correct location within a Textbox or Input object when a vector font is used. (MicroHelp)

Windows controls how text is rotated, and appears to have some problems doing so. Some values of CaptionRotation may produce unusual angles. (Microsoft)

Even for vector fonts, Windows seems to have trouble rotating smoothly through the 0 to 360 degree range of the LetterRotation attribute. (Microsoft)

For Combo and Listbox objects, after adding/changing items at run time, we recommend that you do not change appearance attributes of the object (such as FontSize); doing so might reset the item list back to its original state. (MicroHelp)

The Line object erases previously drawn Xgraphics located within a rectangular area bounded by the Line. If this creates a problem for your project, employ the LINE command instead. (Microsoft)

Due to a Windows bug, we do not recommend using a Menu object when the TitleBar attribute of the Layout object is set to No. (Microsoft)

The MaxDrop attribute might not produce the correct number of lines. (MicroHelp)

PAINT has been known to fail on certain display adapters. (Microsoft)

RBOX does not work well when the box size is small. (Microsoft)

Do not set SetFocus to 1 while processing a ChangeEvent. (Microsoft)

STYLE (2, 7) has been known to produce invisible output on certain computers. (Microsoft)

Displaying .WMF files that contain color palettes can cause an unrecoverable reduction in available Windows resources. You may be able to correct the problem by resaving the .WMF file (the graphics utility named HiJaak seems to the job). (Microsoft)

Pic images are not scaled on Frame objects. (MicroHelp)

If the Caption of a Frame object contains digits separated by spaces, the Caption might not be displayed correctly. (MicroHelp)

Due to a bug in Windows, printing pages via ext(19) or ext(119) might fail if Windows is running in greater than 256 colors. (Microsoft)

Due to a bug in Windows, the mouse cursor might change to a "not allowed" symbol when dragging over certain classes of objects. (Microsoft)

During DDE communications (such as via the Dde() function or links with external snap-on components), Windows might briefly change the mouse cursor to an hourglass. (Microsoft)

When a user resumes at a bookmark, if AutoRedraw is enabled, Windows may only be able to restore a

partial copy of a window's background. (Microsoft)

If an external add-on GPFs unexpectedly upon certain keypresses, you may need to add "dummy" empty VB forms to the project. This is due to a bug in Windows. Try adding enough empty forms to have either 32, 40, 48 or 56 forms total in the project and retry. (Microsoft)

Shape and Line objects do not plot in the correct location when a window is scrolled via Scrollable. (Crescent)

Appendix E - Tranferring Visual Basic Skills

If you are familiar with Microsoft's Visual Basic (VB) programming language, the following information may be of assistance when using Everest. Listed below are many elements of Visual Basic, along with their Everest A-pex3 equivalents. Note that VB properties (the largest category of key words) are NOT included in the list below because most have similarly (if not identically) named attributes in Everest. The same is true for VB events.

Visual Basic	Everest Equivalent
Abs()	Abs()
AppActivate	Shl()
And	&
Asc()	Asc()
Atn()	Atn()
Beep	Mbx()
Call	CALL
Chr()	Chr()
Choose()	Pik()
Circle	CIRCLE
Close	Fyl(0)
Cls	ERASE
Command	Ext(43)
Controls	Obj()
Cos()	Cos()
CurDir()	Ext(0)
Date	Dat()
Day	Dat(2)
Debug	DPRINT
Dim	DIM
Dir	Ext(41), Ext(42)
Do	DO
Do While	DO IF
DoEvents	DoEvents, Ext(101)
Elseif	ELSEIF
End	BRANCH @end
End If	ENDIF
EndDoc	LPRINT
Environ	Env()
EOF	Fyl(-11)
Erase	DELVAR
Error	Ext(110)
Error$	Msg()
Exit Do	OUTLOOP
FileCopy	Fyl(-5)
FileDate	Fyl(-3)
FileLen	Fyl(-2)
FillStyle	STYLE
Fonts	Fnt()
For...Next	DO
Form	window

Format()	Fmt()
Get	Fyl(14)
GetText	Ext(107)
Gosub	GOSUB
GoTo	GOTO
Hex()	Hex()
Hide	Visible
Hour	Tim(1)
If	IF
Input()	Fyl(11)
InputBox()	Ibx()
Instr()	* operator
Int()	Lwr()
Kill	Fyl(31)
LCase()	Lwr()
Left()	\ operator
Len()	Len()
Let	= operator
Like	=P= operator
Line	BOX, FBOX, LINE
Line Input #	Fyl(11)
LinkExecute	Dde()
LinkPoke	Dde()
LinkRequest	Dde()
LinkSend	Dde()
Load	Create
LoadPicture()	BgndPicture, PictureFile
Log()	Log()
Loop	LOOP
Loop While	LOOP IF
LTrim()	Ltr()
Me	window(0)
Mid	^ operator
Mid()	Mid(), ^^ operator
Minute	Tim(2)
MkDir	Fyl(-12)
Mod	^/ operator
Month	Dat(1)
MousePointer	Mse()
Move	Move
MsgBox()	Mbx()
NewPage	LPRINT
Now	Dat(6)
Open	Fyl()
Or	@
Parsing strings	Operators
Point	Ext(1)
PopupMenu	PopupMenu
Print	PRINT
Print #	Fyl(22), Fyl(23)
PrintForm	Ext(19)

PSet	POINT
Put	Fyl(24)
ReDim Preserve	REDIM
Refresh	Update
Rem	$$
Return	GOTO @exitprog
Rgb()	Rgb()
Right()	/ operator
Rnd	Rnd()
RTrim() Rtr()	
Scale	SCALE
Second	Tim(3)
Seek	Fyl(11)
SendKeys	Key(7)
SetFocus	SetFocus
SetText	Ext(108)
Shell()	Shl()
Show	OPEN, Visible
Sin()	Sin()
Space()	$ operator
Sqr()	Sqr()
Stop	STEP
String()	$ operator
Switch()	Pik()
Tan()	Tan()
TextWidth	Ext(109)
Time	Tim()
Timer	Tim(0)
UBound()	Arr()
UCase()	Upr()
Unload	Destroy
Val()	Val()
VarType	Typ()
Weekday	Dat(5)
While	DO
Wend	LOOP
Year	Dat(3)
ZOrder	ZOrder

Appendix F - File Handling

An important feature of Everest is its ability to import and display files of various types, for example, images stored in .BMP and .PCX files. At design time, you typically specify the name of the desired file. For example, in the case of the Picture object's PictureFile attribute, you might enter `cat.pcx`.

EXTERNAL FILES

Everest uses the term "external file" to refer to any file that is not stored within the book. In the PictureFile example above, CAT.PCX is an external file; PictureFile is simply pointing to it by name. When you distribute your project to others, you must provide the book plus any external files it employs. Everest's Project Packager utility can automatically gather together all referenced external files for you.

External File Locations

When you do not specify the location (drive, subdirectory, web site, etc.) of an external file, Everest assumes it is in the same place as the book. In the PictureFile example above, Everest will attempt to load CAT.PCX from the same location as the book (i.e. wherever the current .ESL file resides).

In general, omitting external file locations is the best approach. This allows the end user to install your project in any location (C drive, D drive, Internet, etc.), and Everest will easily find all the external files.

Everest lets you set PictureFile to, for example, `C:\animals\cat.pcx`. That may work fine on your computer, but will not on the end user's computer, unless they also have a C:\animals subdirectory containing a CAT.PCX file. You certainly could create such a subdirectory on the end user's computer when your project is installed, but that is not a flexible approach. Most of the time, it is best to not specify a file location.

Choosing a File

As you probably know, while authoring, you can double click on PictureFile and Everest will display the Load File window. The Load File window lets you browse your disks to find the desired file. When you find it, you can double click on the file name.

If the file you choose is not in the same location as the book, Everest will prefix the file name with the location. As described above, this is usually undesirable. Here's the easy solution:

1) using the Load File window, find the file you want
2) if the file is not in the book's location, click on the Copy File button
3) copy the file into the book's location
4) double click on the copied file

INTER/INTRANET FILES

If your book will not be run from the Inter/intranet, but you wish to load a specific external file from an Inter/intranet site, simply prefix the file name with the site address. Be sure to start the address with `http://`. So, your entry for an atttribute such as SPictureFile might resemble:
`http://www.xyz.com/cat.pcx`.

Alternatively, it is often easier to use the special symbol "~:" in place of the site address. For example,

`~:cat.pcx`. Doing so tells Everest to employ the location you specify in the INetSite entry in the EVEREST.INI file. You can change the INetSite entry via the Settings window.

When it downloads a file from the Inter/intranet, Everest first stores it in the location you can specify in the CachePath entry in the EVEREST.INI, then loads it into your project. This location should be a fast disk drive on your local computer.

If your book and graphics files will all be stored (and run from) the Inter/intranet, you need not specify the site address each time you reference a file in an SPictureFile (or similar) attribute. This is because if the book (the .ESL file) is on the Inter/intranet, Everest will automatically look for external files there as well.

NOTE: The Inter/intranet related features in Everest require that the computer is already set up and configured for net access. Basically, if the computer can access the net via software such as Navigator or Internet Explorer, then Everest should also be able to do so. Everest communicates with the net via the computer's copy of WINSOCK.DLL.

ZIP FILES

At run time, if Everest cannot find an external file, it will automatically check for a file of the same name, except with a .ZIP extension. For example, if CAT.PCX was not found, Everest automatically then checks for a CAT.ZIP file. If such a .ZIP file is found, Everest automatically uncompresses it, and re-attempts to load the file of the original name. The uncompression is performed to the temporary subdirectory you specify via the TempPath entry in the EVEREST.INI.

This automatic ZIP check is especially beneficial if your project is stored on a Inter/intranet site. Since .ZIP files are typically smaller than their un-zipped couterparts, Everest can download them faster (sometimes MUCH faster). To prepare your project to take advantage of this feature, simply enable the "Compress each file into ZIP format" option in Everest's Project Packager utility.

The auto-ZIP check feature is available for the following attributes: AnimFile, BgndPicture, BMPFile, FileName, Pic, PictureFile and SPictureFile. If, for special purposes, you want to disable Everest's auto-ZIP check feature, add 2 to the value of Sysvar(187) at the start of your project.

EMBEDDING FILES

External files are just that: external to the book. Alternatively, Everest lets you copy a file directly into the book; such a file is called "embedded." If a file is embedded in the book, when you distribute your project, it is not necessary to also distribute the external file.

To tell Everest to embed a file into the book:

1) set the PictureFile (or other attribute that allows external files) to the desired external file
2) from the Options menu, choose Embed File
3) the Load File dialog appears; find and reselect the file to embed (doing so verifies which file you want to embed)

When successful, Everest will tell you the size of the external file, as well as the size when embedded. Everest compresses embedded files, some up to a 100:1 ratio. You can recognize an embedded file by the | character that prefixes the file name in the attribute. In fact, you can manually add the | prefix instead of using the Embed File menu item.

Embedded File Advantages and Disadvantages

Embedded files have both advantages and disadvantages compared to external files. The advantages:

1) embedded files are compressed, thereby saving disk space
2) they afford a simple means of copy protection

The disadvantages:

1) if you subsequently modify the original external file (such as with a graphics editor), it must be re-embedded into the book
2) at run time, Everest must copy the embedded file from the book to a temporary location, then load it normally; that extra step makes the process a bit slower
3) they make the book larger

At run time, Everest copies the embedded file from the book to a temporary location. This location is determined as follows:

1) the location specified in Sysvar(152); upon start up Sysvar(152) is set to the value of the TempPath entry in the EVEREST.INI
2) if Sysvar(152) is empty, then the location specified by the DOS environment variable TEMP
3) if TEMP is empty, then the location specified by Sysvar(16), that is, the location of the book

The temporary location must have enough disk space to handle all the embedded files likely to be used by one page.

Choosing an Embedded File

If you want to use a previously embedded file, open the Load File dialog, click on the Embedded button, and choose from the list displayed.

Managing Embedded Files

If you have several files embedded in the book, you can manage them (freshen, delete, etc.) with help from the Embed Manager found on the Utilities menu.

SPECIAL LOCATIONS

For special purposes, you may want to load a file from a location other than the book. For example, you may want to store your .WAV files in a subdirectory one level deeper than the book. Everest offers several special symbols and features you can employ to help you keep your project "location independent" (so the end user can install it anywhere). These symbols are best described via example:

Example	**Load From Location**
`cat.pcx`	the same location as the book
`animals\cat.pcx`	same location, except in the subdirectory within named animals (i.e. one directory level deeper than the book)

`|cat.pcx`	embedded within the book
`|C:\animals\cat.pcx`	embedded within the book (the path is ignored)
`C:\cat.pcx`	C:\ (root directory of C: drive)
`C:cat.pcx`	current default directory of C: drive
`C:\animals\cat.pcx`	C:\animals
`?:cat.pcx`	the same location as the book
`?:\animals\cat.pcx`	same drive as the book, but in the \animals subdirectory
`@:cat.pcx`	current DOS default drive and subdirectory
`&:cat.pcx`	the Windows directory (often C:\WINDOWS)
`&:\animals\cat.pcx`	the \animals subdirectory on the drive that contains Windows
`%:cat.pcx`	the location of the current Everest .EXE program
`%:\animals\cat.pcx`	the \animals subdirectory on the drive that contains the current Everest .EXE program
`%:animals\cat.pcx`	same location as the current Everest .EXE program, except in the nested subdirectory named animals
`^:cat.pcx`	the location of the EVEREST.INI file
`*:cat.pcx`	the location specified via the StarPath entry in the EVEREST.INI
`http://www.xyz.com/cat.pcx`	the Internet site www.xyz.com
`~:cat.pcx`	the location specified via the INetSite entry in the EVEREST.INI

OTHER NOTES

For special purposes, you can extract embedded files from the book at run time via the Fyl(-9) function.