

Annotation ActiveX Control

User's Guide

IMAGE  *BASIC*

Diamond Head Software, Inc.
1217 Digital Drive Ste. 125
Richardson, Texas 75081
(972) 479-9205

COPYRIGHT NOTICES

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Diamond Head Software, Inc., except in the manner described in the documentation.

This software product contains proprietary software components developed by a number of different software companies, referred herein as "Third Party Licensors". This documentation and the software that you purchased are protected by one or more of the following copyright notices:

Copyright © 1996, 1997 Diamond Head Software, Inc. All rights reserved.

Company and product names mentioned in this documentation are trademarks or registered trademarks of their respective companies. Lotus and Lotus Notes are registered trademarks of Lotus Development Corporation. Windows is a trademark and Microsoft is a registered trademark of Microsoft Corporation.

DIAMOND HEAD SOFTWARE INC. AND ITS THIRD PARTY LICENSORS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SOFTWARE. DIAMOND HEAD SOFTWARE, INC. AND ITS THIRD PARTY LICENSORS DO NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME JURISDICTIONS. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

IN NO EVENT WILL DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS AND/OR THEIR DIRECTORS, OFFICERS, EMPLOYEES OR AGENTS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU. Diamond Head Software Inc.'s and its Third Party Licensors' liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to \$50.

Contents

Chapter 1 : Getting Started	1
Introduction to the Annotation Control.....	1
Displaying Annotations on an Image.....	2
Data Management in Memory.....	3
Licensing Configuration and Verification	4
Chapter 2 : Using the Annotation Control	7
Definitions of the Annotation Types.....	7
Arrow Annotation Class.....	8
Bitmap Annotation Class.....	10
Freehand Annotation Class.....	12
Highlight Annotation Class.....	14
Line Annotation Class.....	15
Redaction Annotation Class.....	17
Pointer Class	19
Sticky Note Annotation Class.....	20
Text Annotation Class.....	21
Creating and Editing Annotations.....	23
Setting Defaults for New Annotations.....	24
Editing Annotations -- Graphical.....	25
Editing Annotations -- Programmatic.....	29
Grouping Annotations.....	36
Removing Annotation Objects.....	38
Viewing and Printing Annotations.....	39
Displaying and Hiding Annotations.....	39
Ordering the Display of Annotations.....	40
Printing Annotations with an Image.....	41
Saving and Loading Annotations.....	42
Saving and Loading Annotations in TIFF Files.....	42
Saving and Loading Annotation Files.....	45
Imprinting Annotations into an Image.....	47
Chapter 3 : Reference	49
Reference List.....	49
Index	95

Chapter 1 : Getting Started

Introduction to the Annotation Control

The Annotation control allows the user to create, edit, save and load eight different types of annotation objects. The following types of annotations are available to the user of ImageBASIC's Annotation control:

Highlight	A transparent rectangle of any color to draw attention to a particular portion of the image
Redaction	An opaque rectangle of any color to conceal a portion of the image
Text	Simple overlay of a text string on the image
Bitmap	Insertion of any Windows bitmap file for display on the image
Sticky Note	Displays a small icon that opens to reveal a Notepad-like form for the storage and transfer of relatively large quantities of text
Line	A single, straight line between any two points on the image
Arrow	A single, straight line between any two points on the image with an arrow head at either or both ends
Freehand	Unrestricted line of any length and curvature

Each individual annotation, for example, a highlighted region or a freehand drawing, is considered a distinct object. A more thorough description of each of these classes of annotation objects, including the descriptive properties that apply to them may be found in the section "Definitions of the Annotation Types" on page 7.

Each annotation object can be created or modified either by the user clicking and dragging with the mouse to describe the object or by the application specifying a series of property values that describe the object. The options for creating and modifying annotation objects may be found in the section "Creating and Editing Annotations" on page 23.

For those applications that may need to display different annotations on a single image depending upon the operator, a set of annotation objects may be grouped together on a single layer. Multiple layers can then be displayed on a single image, and each layer may be independently displayed or hidden. The process of creating and displaying layers is described in the section titled "Grouping Annotations" on page 36.

Displaying Annotations on an Image

Because the Annotation control does not have display capabilities, each ImageBASIC Annotation control must be told which Display control will show its annotation objects. The Display control that will be used must be specified in the Annotation control's **DisplaySource** property.

- The **DisplaySource** property may specify any ImageBASIC Display control.
- The named Display control will show all annotation objects that are loaded or created by the Annotation control.

For example, suppose that you have an image retrieval and display application that uses the following ImageBASIC controls:

One TMSSequoia File control named *TMSFile1*

One TMSSequoia Display control named *TMSDispl*

One Annotation named *Annotel*

If

- 1) the File control named *TMSFile1* is to retrieve image files from a network drive, and
- 2) the Display control named *TMSDispl* is to show the retrieved images, and
- 3) annotations are to be created or modified by the Annotation control named *Annotel*,

the following links must be made between the ImageBASIC controls before any images or annotations may be displayed:

```
' the Display control will receive image data from
' the File control
TMSDispl.ImageDataSource = TMSFile1.Link

' annotations will be shown on the Display control
Annotel.DisplaySource = TMSDispl.Link
```

Data Management in Memory

Each image page and any additional information associated with that page are managed within ImageBASIC as a single structure. All information that is maintained in addition to the image itself is referred to as sideband data. Sideband data can include annotations along with other information such as TIFF tag values.

When an image is loaded by a File control or created by a Scan control, the image structure is built for each page. This entire image structure is available to each ImageBASIC control that is linked to this source control.

For example, when a File control loads a TIFF file, if there are ImageBASIC annotations in the file's header, those annotations are also loaded and stored in the image's sideband data.

- When a Display control is linked to the File control, the entire image structure is received by the Display. This structure includes the annotation sideband data.
- Any recipient control, such as the Display itself, that does not understand the annotation sideband data will simply ignore it.
- If an Annotation control is linked to this Display control, the annotations in the sideband data will be read and displayed. Any changes made to the annotations will be held with the image that the Display is showing.

Any change to either the image or the sideband data is considered a new image page. Therefore, when a Display control accepts a page from an File control, and the Annotation control is used to add annotations to that page, the Display control is now holding a page that is different from the one held by the File control.

- To save the new annotations, the page held by the File control must be replaced with the page held by the Display control.

For this purpose, the File control has a method called **ReplacePage**. The **ReplacePage** method replaces the old page in the document with the new one that has the annotations.

The File control's document has now changed and should be saved as any other changed image.

- Alternatively, the File control can read the image page directly from the Display control and save that image page to file without modifying the current document. Refer to the File control's User's Guide for more information.

Licensing Configuration and Verification

In order to run, each ImageBASIC control must be able to verify the presence of a valid license token. These license tokens are stored on either a hardware key (shipped with each toolkit) or in a licensing database (the standard runtime distribution format).

Utilizing these tokens, licensing in ImageBASIC is based on enabling a set number of concurrent seats. Each license token allows a single PC to run any number of instances of a single control. For example, a single token for a Display will allow one workstation to concurrently run multiple applications, each of which employs any number of Display controls.

A unique token is required for each different type of ImageBASIC component that you have licensed:

- There is a special type of token for the TMSSequoia Display control, another for the TextBridge control, another for the ScanFix control, and yet more token types for each additional control. The 16-bit and 32-bit versions of each control are also licensed separately.
- Each one of these licenses also comes in two varieties: *runtime* and *development*.

As suggested by the names, *runtime* licenses are necessary for an executable to function, and *development* licenses are necessary to develop an application using ImageBASIC.

A design time license will function as a runtime license, removing the need to add runtime tokens for application testing during development.

Where the Licenses are Kept

ImageBASIC will find licenses stored in either one of two locations -- in a licensing database or on a hardware key.

- The hardware key is plugged into a parallel port on the computer using ImageBASIC and will be automatically found each time an ImageBASIC component is used.

Note: When using Windows NT, the parallel port must be configured using the Sentinel drivers that are shipped with ImageBASIC.

- The licensing database must be created on the site where it will be used and may not be moved from the location in which it is installed.

The inability to move a licensing database is one of its protection features. This attachment to a single location is formed when the

licensing database is first created. Although it may not be moved once created, the licensing database can be written to a network drive to which all the machines running ImageBASIC have access.

A file called IMGBASIC.INI must be on each of these networked machines. This file contains an entry pointing to the location of the licensing database. ImageBASIC can now search the licensing database for an available copy of each license it needs to run.

The licensing database may also be created on a local drive, activating ImageBASIC on only that one machine. The same INI file is still necessary to pinpoint the location of the database.

How the License Tokens are Used

As each application that uses ImageBASIC is initiated, or the control is loaded into the development environment, the ImageBASIC licensing server attempts to find the proper token for each component.

For example, if an executable has been developed that uses ImageBASIC to display existing images, and then calls on TextBridge to perform OCR on that image, that application must be able to find one *Display runtime license*, one *File runtime license* and one *TextBridge runtime license*.

- If the application is successful in finding these licenses, it will load normally and function exactly as it was programmed.

When the application finds these licenses and is thereby informed that the correct licenses are available, it simultaneously locks the licenses so that no other application can use the same licenses to run at the same time.

If two copies of these same licenses are available, another computer will be able to run the same application and will then lock the second license.

- If the application cannot verify the presence of the required tokens, the ImageBASIC components will not operate.

The **Active** property of the controls that did not find license tokens will be set to False. The state of the **Active** property can be verified during Form Load.

A runtime error that can be trapped during Form Load will also occur.

Licensing Token Release

When an application that has locked one or more tokens terminates normally, the tokens are released and can immediately be taken by another user if a network licensing database is being used. This is the process by which concurrent licensing for any number of seats may be enabled.

- If the application ends abnormally -- the user might reboot or a concurrently running Windows application might lock up -- then the release of the licenses is conditional on the network or disk operating system.
- For Novell networks, the default time for releasing a file lock after a connection is lost is 5 (five) minutes.

For other networks, your system administrator should be able to tell you how long the NOS takes to release the lock.

The system administrator should be able to change the default release time to satisfy your particular needs.

- If the licensing database has been installed on a stand-alone machine, the locks are immediately released and will again be available when the application is started again.

Chapter 2 : Using the Annotation Control

Definitions of the Annotation Types

The following types of annotation objects may be created by the Annotation control:

Highlight	A transparent rectangle of any color to draw attention to a particular portion of the image.
Redaction	An opaque rectangle of any color to conceal a portion of the image.
Text	Simple overlay of a text string on the image.
Bitmap	Insertion of any Windows bitmap file for display on the image.
Sticky Note	Displays a small icon that opens to reveal a Notepad-like form for the storage and transfer of relatively large quantities of text.
Line	A single, straight line between any two points on the image.
Arrow	A single, straight line between any two points on the image with an arrow head at either or both ends.
Freehand	Unrestricted line of any length and curvature.

The following sections provide a more thorough description of each of the type of annotations. Also listed in each class description are the properties that describe an annotation object of this type. The programmatic creation or modification of an annotation object is performed through these properties.

After each type of annotation is described, the process of creating annotations is described. Because annotations may be added to or removed from the display by two nearly exclusive techniques, the description of annotation creation is divided into two main sections -- creating annotations through the graphical interface and creating annotations through the programmatic interface.

After the descriptions of creating and modifying annotations, the processes of saving and loading annotations is described.

Arrow Annotation Class

Arrow annotation objects are straight lines with an arrow head drawn at either or both ends.

Arrows may be created through a graphical interface by setting the **LeftButtonOption** or **RightButtonOption** property to *10--Arrow*.

- When the appropriate mouse button is depressed over the Annotation control's display window, as specified in the **DisplaySource** property, a new Arrow object is created and the **NoteCreated** event occurs.
- By holding down the mouse button and moving the mouse cursor, the second end point of the object can be defined. When the mouse button is released, the arrow is drawn on the image.
- Attributes of the new Arrow may be changed in the **NoteCreated** event by modifying the descriptive properties listed in the paragraphs below.

Selecting Line Style and Size

The line size is specified through the **NoteThickness** property. This integer property specifies the number of image pixels across the width of the line that is drawn.

The style of line that is drawn (solid, dotted, etc.) is specified through the **NoteLineStyle** property. This enumerated property is valid only when **NoteThickness** is 1 (one). Available options are as follows:

0	Solid	—————
1	Dash	-----
2	Dot
3	Dash Dot	- - - - -
4	Dash Dot Dot	- . - . - . -

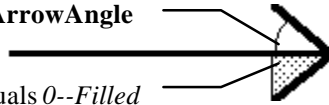
Selecting Arrowhead Position and Style

The following properties define the arrowhead that is drawn on the Arrow object:

NoteArrowLocation	Enumerated property defining the position of the arrow head on the object. The valid options are as follows:
0	Head (Default)
1	Tail
2	Both

NoteArrowAngle Specifies the angle between the main line of the Arrow object and the sides of the arrow head. May be set up to 180, which would create a straight line.

Angle specified by **NoteArrowAngle**



Area filled if **NoteArrowStyle** equals 0--*Filled*

NoteArrowLength Specifies the pixel length of each side of the arrowhead.

NoteArrowStyle Enumerated property specifying whether or not the arrowhead is drawn solid or hollow. Set according to the following options:

- 0 Filled (Default)
- 1 Hollow

Selecting Line Color

The color of the Arrow object is specified through the **NoteColor** property. This property accepts an RGB value such as is returned from Visual Basic's RGB function in which the level of each individual color component is defined:

```
Annotel1.NoteColor = RGB(200, 0, 200)
```

The valid range for a normal RGB color is 0 to 16,777,215 (FFFFFF hex).

Properties Describing Arrow Class Objects

The following properties describe each Arrow class object. Refer to 'Editing Annotations -- Programmatic' on page 29 for details on these properties.

- These properties may be used to modify an object of this type.
- Many of these properties may be set to default values by first setting the **NoteIndex** property to 0 (zero) and the **DefaultClass** property to 7--*Arrow*.
- Any changes made in these or any other descriptive properties while **NoteIndex** equals 0 will be reflected in all future objects of the type specified in **DefaultClass**

Arrow Class Descriptive Properties	Standard Descriptive Properties
NoteArrowAngle	NoteBottom
NoteArrowLocation	NoteClass
NoteArrowLength	NoteColor
NoteArrowStyle	NoteLeft
NoteLineStyle	NoteName
NotePointCount	NoteRight
NotePointIndex	NoteSelected
NotePointX	NoteTop
NotePointY	NoteVisible
NoteThickness	NotePrint

Bitmap Annotation Class

The Bitmap class annotation object inserts any Windows BMP file as a new annotation object. The file may be scaled to fit an arbitrary rectangular region or may be displayed at its original resolution. When this object is saved, the BMP file is saved as a DIB included in the save file.

A new Bitmap object may be created graphically by setting the **LeftButtonOption** or **RightButtonOption** property to *6--Bitmap*. The file to display may be specified before the object is created by setting the default value for the **NoteFileName** property.

Selecting a File

The BMP file to load as an Bitmap object may be specified either by setting a property or calling a file browse dialog:

- The **NoteFileName** property specifies the fully qualified path and file name of the Windows BMP file that is displayed.
- The **ShowFileDialog** method loads a standard file selection dialog, allowing the user to browse or specify the file name. The annotation object that is specified in the **NoteIndex** property will display this file.

Scaling the Bitmap for Display

When the BMP is displayed, the **NoteDrawMode** property specifies the scaling performed on the file. This is an enumerated property with the following options:

- 0 Fit Region to BMP (Default)
- 1 Fit BMP to Region
- 2 Clip BMP to Region

0--Fit Region to BMP will display the bitmap scaled to the same factor as the underlying image. If the bitmap extends beyond the boundaries of the underlying image, the excess will be truncated.

1--Fit BMP to Region scales the bitmap to fit in the region that is defined either by selecting the object with the mouse and sizing it with the grab handles or by setting the **NoteTop**, **NoteRight**, **NoteBottom**, and **NoteLeft** properties.

2--Clip BMP to Region will fit as much of the bitmap in the defined region as possible and truncate any portion that extends beyond the region's boundaries.

Properties Describing Bitmap Class Objects

The following properties describe each Bitmap class object. Refer to "Editing Annotations -- Programmatic" on page 29 for details on these properties.

- These properties may be used to modify an object of this type.
- Any of these properties may be set to default values by first setting the **NoteIndex** property to 0 and setting **DefaultClass** to *4--Bitmap*. Any changes made in these or any other descriptive properties will be reflected in all future objects.
- The following table shows the properties that describe Bitmap class annotation objects.

Bitmap Class Descriptive Properties	Standard Descriptive Properties
NoteBitmapImageHeight	NoteBottom
NoteBitmapImageWidth	NoteClass
NoteBitmapResolution	NoteColor
NoteDrawMode	NoteLeft
NoteFileName	NoteName
NoteFillStyle	NoteRight
NotePicture	NoteSelected
NoteRotation	NoteTop
	NoteVisible
	NotePrint

Freehand Annotation Class

The Freehand class allows the user to draw a line of any length or shape. After engaging the Freehand mode (refer to the **LeftButtonOption** and **RightButtonOption** properties), begin free style drawing by moving the mouse pointer to a point on the image, depress the mouse button and move the mouse about. Each time the mouse button is released and depressed, a new annotation object is begun, an important issue when deleting or editing objects.

Selecting Line Style and Size

The line size is specified through the **NoteThickness** property. This integer property specifies the number of image pixels across the width of the line that is drawn.

The style of line that is drawn (solid, dotted, etc.) is specified through the **NoteLineStyle** property. This property is valid only when the line is a single pixel wide; i.e., when the **NoteThickness** property equals 1 (one).

This is an enumerated property with the following values:

- 0 Solid
- 1 Dash
- 2 Dot
- 3 Dash Dot
- 4 Dash Dot Dot

Selecting Line Color

The color of the Freehand object is specified through the **NoteColor** property. This property accepts an RGB value such as is returned from Visual Basic's RGB function in which the level of each individual color component is defined:

```
Annotel.NoteColor = RGB(200, 0, 200)
```

The valid range for a normal RGB color is 0 to 16,777,215 (FFFFFF hex).

Properties Describing Freehand Class Objects

The following properties describe each Freehand class object. Refer to 'Editing Annotations -- Programmatic' on page 29 for details on the use of these properties.

- These properties may be used to modify an object of this type.
- Many of these properties may be set to defaults by setting the **NoteIndex** property to 0 (zero) and the **DefaultClass** property to *8--Freehand*.
- Any changes made in these or any other descriptive properties while **NoteIndex** equals 0 (zero) will be reflected in all future objects of the type specified in **DefaultClass**

Freehand Class Descriptive Properties	Standard Descriptive Properties
NoteLineStyle	NoteBottom
NotePointCount	NoteClass
NotePointIndex	NoteColor
NotePointX	NoteLeft
NotePointY	NoteName
NoteThickness	NoteRight
	NoteSelected
	NoteTop
	NoteVisible
	NotePrint

Highlight Annotation Class

To highlight a rectangular region in the image, select the Highlight class. The Annotation engine may be placed in this mode by setting the **LeftButtonOption** or **RightButtonOption** property to *3--Highlight*. Highlight a region by placing the mouse cursor to the top, left corner of the region and then moving the cursor to the opposite corner while holding down the mouse button.

Selecting Highlight Color

The color of the Highlight object is specified through the **NoteColor** property when the **NoteIndex** property is set to select a Highlight object. **NoteColor** accepts an RGB value such as is returned from Visual Basic's RGB function:

```
Annot1.NoteColor = RGB(200, 0, 200)
```

The valid range for a normal RGB color is 0 to 16,777,215 (FFFFFF hex).

Selecting Highlight Shape

The default shape for a highlighted region is a rectangle, but this shape may be modified through the **NoteShape** property. When the **NoteIndex** property is set to select a Highlight object or to set defaults, the **NoteShape** property may be set according to the following enumerated options:

- 0 Rectangle (Default)
- 1 Rounded Rectangle
- 2 Ellipse

Selecting Highlight Pattern

When the Highlight object is displayed, the fill color may be solid or may be drawn in another pattern. When the **NoteIndex** property is set to select a Highlight object or to set default values, the **NoteFillPattern** may be set according to the following enumerated options:

- 0 Hollow
- 1 Solid (Default)
- 2 Horizontal Lines
- 3 Vertical Lines
- 4 Forward Diagonals
- 5 Backward Diagonals
- 6 Grid Pattern
- 7 Diamond Pattern

Properties Describing Highlight Class Objects

The following properties describe each Highlight class object. Refer to "Editing Annotations -- Programmatic" on page 29 for details on the use of these properties.

- These properties may be used to modify an object of this type.
- Many of these properties may be set to default values by first setting the **NoteIndex** property to 0 and the **DefaultClass** property to *Highlight*.
- Any changes made in these or any other descriptive properties while **NoteIndex** equals 0 will be reflected in all future objects of the type specified in **DefaultClass**

Highlight Class Descriptive Properties	Standard Descriptive Properties
NoteFillPattern	NoteBottom
NoteFillStyle	NoteClass
NoteShape	NoteColor
	NoteLeft
	NoteName
	NotePrint
	NoteRight
	NoteSave
	NoteSelected
	NoteTop
	NoteVisible

Line Annotation Class

To draw a simple, straight line between two points on the image, select the Line class of annotation object. The line thickness, color, style, and transparency are all fully configurable as described below.

Selecting Line Style and Size

The line size is specified through the **NoteThickness** property. This integer property specifies the number of pixels across the width of the line that is drawn.

The style of line that is drawn (solid, dotted, etc.) is specified through the **NoteLineStyle** property. This is an enumerated property with the following values:

- 0 Solid
- 1 Dash
- 2 Dot
- 3 Dash Dot
- 4 Dash Dot Dot

Selecting Line Color

The color of the Arrow object is specified through the **NoteColor** property. This property accepts an RGB value such as is returned from Visual Basic's RGB function in which the level of each individual color component is defined:

```
Annot1.NoteColor = RGB(200, 0, 200)
```

The valid range for a normal RGB color is 0 to 16,777,215 (FFFFFF hex).

Properties Describing Line Class Objects

The following properties describe each Line class object. Refer to "Editing Annotations -- Programmatic" on page 29 for details on the use of these properties.

- These properties may be used to modify an object of this type.
- Many of these properties may be set to default values by first setting the **NoteIndex** property to 0 and the **DefaultClass** property to *6--Line*.
- Any changes made in these or any other descriptive properties while **NoteIndex** equals 0 will be reflected in all future objects of the type specified in **DefaultClass**

Line Class Descriptive Properties	Standard Descriptive Properties
NoteFillStyle	NoteBottom
NoteLineStyle	NoteClass
NotePointCount	NoteColor
NotePointIndex	NoteLeft
NotePointX	NoteName
NotePointY	NotePrint
NoteThickness	NoteRight
	NoteSave
	NoteSelected
	NoteTop
	NoteVisible

Redaction Annotation Class

Similar to Highlighting, the Redaction class allows the user to draw a colored rectangle over a portion of the image. Unlike highlighting, the portion of image data under the Redaction region is not visible.

Selecting Redaction Color

The color of the Redaction object is specified through the **NoteColor** property when the **NoteIndex** property is set to select a Redaction object or set to define default values. **NoteColor** accepts an RGB value such as is returned from Visual Basic's RGB function:

```
Annotation1.NoteColor = RGB(200, 0, 200)
```

The valid range for a normal RGB color is 0 to 16,777,215 (FFFFFF hex).

Selecting Redaction Shape

The default shape for a redaction region is a rectangle, but this shape may be modified through the **NoteShape** property. When the **NoteIndex** property is set to select a Highlight object or to set defaults, the **NoteShape** property may be set according to the following options:

- 0 Rectangle (Default)
- 1 Rounded Rectangle
- 2 Ellipse

Selecting Redaction Pattern

When the Redaction object is displayed, the fill color may be solid or may be drawn in another pattern. When the **NoteIndex** property is set to select a Redaction object or to set default values, the **NoteFillPattern** may be set according to the following options:

- 0 Hollow
- 1 Solid (Default)
- 2 Horizontal Lines
- 3 Vertical Lines
- 4 Forward Diagonals
- 5 Backward Diagonals
- 6 Grid Pattern
- 7 Diamond Pattern

Note that all fill patterns will completely obscure any image data below the Redaction object. When an option other than *--Solid* is selected, the area between the hatch lines will be white.

Properties Describing Redaction Class Objects

The following properties describe each Redaction class object.

The values of these properties may be used to create or modify an object of this type. Refer to "Editing Annotations -- Programmatic" on page 29 for details on the use of these properties.

- These properties may be used to modify an object of this type.
- Many of these properties may be set to default values by first setting the **NoteIndex** property to 0 and the **DefaultClass** property to *--Redaction*.
- Any changes made in these or any other descriptive properties while **NoteIndex** equals 0 will be reflected in all future objects of the type specified in **DefaultClass**

Redaction Class Descriptive Properties	Standard Descriptive Properties
NoteFillPattern	NoteBottom
NoteFillStyle	NoteClass
NoteShape	NoteColor
	NoteLeft
	NoteName
	NotePrint
	NoteRight
	NoteSave
	NoteSelected
	NoteTop
	NoteVisible

Pointer Class

Selecting the Pointer class allows the user to move, resize, or delete any annotation objects currently visible. Click once on any annotation object to select it.

Moving or Sizing an Annotation Object

Any selected object may be moved by dragging it to the new position with the mouse.

The selected object can be sized by dragging from one of the small squares displayed at the corner of the rectangle outlining the annotation object.

Deleting an Annotation Object

A single annotation object is deleted by specifying the object in the **NoteIndex** or **NoteName** property and executing the **DeleteNote** method. While in the Pointer mode, the **Click** event will occur for each click on an annotation object, and the **NoteIndex** property may be set in this event.

Sticky Note Annotation Class

Selecting the Sticky Note class allows the addition of a text file as an attachment to the image file. To add the text file, select the Sticky Note mode by setting the **LeftButtonOption** or **RightButtonOption** property. Click the mouse pointer in the image where you would like the activating icon to appear. This will open a dialog box where the user may enter any text.

Adding text in this fashion causes an icon to appear on the image when it is viewed in any Image Display Window with an Annotation control when the **ShowNotes** property is set to True. The user must have the Annotation control in this class and click on this icon to open a small window which displays the text. This icon may be deleted just as any other annotation object, and the contents will be lost unless previously saved.

Properties Describing Sticky Note Class Objects

The following properties describe each Sticky Note class object. The values of these properties may be used to create or modify an object of this type. Refer to "Editing Annotations -- Programmatic" on page 29 for details on the use of these properties.

- These properties may be used to modify an object of this type.
- Many of these properties may be set to default values by first setting the **NoteIndex** property to 0 and the **DefaultClass** property to 5--
Sticky Note.
- Any changes made in these or any other descriptive properties while **NoteIndex** equals 0 will be reflected in all future objects of the type specified in **DefaultClass**

Sticky Note Class Descriptive Properties	Standard Descriptive Properties
NoteDrawMode	NoteBottom
NoteFileName	NoteClass
NoteFillStyle	NoteColor
NoteText	NoteLeft
NoteBitmapResolution	NoteName
NotePicture	NotePrint
	NoteRight
	NoteSave
	NoteSelected
	NoteTop
	NoteVisible

Text Annotation Class

The Text class displays any text string directly on top of the displayed image. The text may be may transparent or opaque, and the color, font type, font size, and font style are all fully adjustable.

Selecting Font Size and Style

The **NoteFont** property specifies the type and size of the text that will be displayed. The values reported in this property always reflect the characteristics of the annotation object specified in the **NoteIndex** property.

The **NoteFont** property is composed of five sub-properties which are referenced as shown here:

```
Annotel.NoteFont.Name = "Courier New"  
Annotel.NoteFont.Size = 10  
Annotel.NoteFont.Bold = False  
Annotel.NoteFont.Italic = True  
Annotel.NoteFont.Strikethrough = False
```

These four sub-properties may be independently set, keeping in mind the following points:

- When selecting the font name, any currently installed Windows font may be specified. If an invalid font name is specified, the default font will be applied.
- If the selected font is not a True Type font, the font size will be limited to the installed options. If the font size is set to an invalid valid, the text will be displayed at the nearest legal point size.
- The defaults for bold, italic and strikethrough are False.

Selecting Text Color

The color of the Text object is specified through the **NoteColor** property when the **NoteIndex** property is set to select a Text object or set to define default values. **NoteColor** accepts an RGB value such as is returned from Visual Basic's RGB function in which the level of each individual color component is defined:

```
Annotel.NoteColor = RGB(200, 0, 200)
```

The valid range for a normal RGB color is 0 to 16,777,215 (FFFFFF hex).

Properties Describing Text Class Objects

The following properties describe each Text class object. Refer to "Editing Annotations -- Programmatic" on page 29 for details on the use of these properties.

- These properties may be used to modify an object of this type.
- Many of these properties may be set to default values by first setting the **NoteIndex** property to 0 and the **DefaultClass** property to *Text*.
- Any changes made in these or any other descriptive properties while **NoteIndex** equals 0 will be reflected in all future objects of the type specified in **DefaultClass**.

Text Class Descriptive Properties

NoteDrawMode

NoteFont

NoteFillStyle

NoteText

NoteRotation

Standard Descriptive Properties

NoteBottom

NoteClass

NoteColor

NoteLeft

NoteName

NotePrint

NoteRight

NoteSave

NoteSelected

NoteTop

NoteVisible

Creating and Editing Annotations

Editing and creating annotations may be accomplished either programmatically or by the user through a graphical interface.

Graphical Editing of Annotations

Setting the **LeftButtonOption** or **RightButtonOption** property to a valid option will initialize the annotation engine and place the application in the specified mode. Typically, the mouse pointer will change to indicate that the annotation engine is active and that a new annotation object will be created when the user next clicks on the image.

The user creates a new annotation object by moving the pointer to the desired location and then clicking and dragging to define the area or shape of the object. For example, if the button option property specifies the Highlight object type, the region that is drawn while dragging will be covered highlighted. If the button option property specifies the Arrow object type, an arrow will be drawn from the point where the button is depressed to the point where it is released.

The **LeftButtonOption** and **RightButtonOption** properties may be independently set according to the following values:

- 1 None
- 2 Edit / Pointer
- 3 Highlight
- 4 Redaction
- 5 Text
- 6 Bitmap
- 7 Sticky Note
- 8 Freehand
- 9 Line
- 10 Arrow

Full details on the graphical creation of annotation objects, including the specification of default values for colors and fonts, may be found in *Editing Annotations -- Graphical* on page 25.

Programmatic Editing of Annotations

Executing one of the following methods will create a new annotation object based on the values of the descriptive properties. Refer to *Definitions of the Annotation Types* on page 7 and *Editing Annotations -- Programmatic* on

page 29 for complete details on the descriptive properties and their implementation.

CreateArrow
CreateBitmap
CreateFreehand
CreateHighlight
CreateLayer
CreateLine
CreateRedact
CreateStickyNote
CreateText

Setting Defaults for New Annotations

Default values for many of the characteristics of new annotation objects may be set. Each class of annotation objects -- Highlight, Text, etc. -- may be assigned a different set of default values.

The steps to set new default values for the creation of annotations is as follows:

- 1) Set the **DefaultClass** property to specify the annotation class for which you are setting a new default. This property is set according to the following enumerated list:
 - 3 Highlight
 - 4 Redaction
 - 5 Text
 - 6 Bitmap
 - 7 Sticky Note
 - 8 Line
 - 9 Arrow
 - 10 Freehand
- 2) Set **NoteIndex** to 0, indicating that a default is being set. No object will be selected.
- 3) Set the appropriate note attribute property to its new default value -- see table, below, for a list of properties that may be set.

The following table list all of the properties whose default values may be changed as shown in the steps above. Any attempt to set a default for a class that does not support that property (e.g., **NoteText** for the Highlight class) will be ignored.

Modifiable Property	Applies to Class
NoteColor	All Classes
NoteDrawMode	Bitmap, Sticky Note
NoteFillPattern	Highlight, Redaction
NoteFillStyle	Bitmap, Highlight, Redaction, Sticky Note, Text
NoteFont	Sticky Note, Text
NoteLineStyle	Arrow, Line, Freehand
NoteShape	Highlight, Redaction
NoteText	Sticky Note, Text
NoteThickness	Arrow, Line, Freehand

Editing Annotations -- Graphical

Any type of annotation object may be created through a graphical interface. Using this graphical interface, annotation objects can be defined and modified using the mouse to position and size the object.

- When the **LeftButtonOption** or **RightButtonOption** property is set to one of the following values, the annotation engine is placed in the specified mode.
- A new annotation object of the specified type will be created each time the appropriate button is depressed. As each new object is created, the **NoteCreated** event occurs. Changes may be made to the new annotation object in this event before the object is displayed.
- Existing annotation objects can be sized or moved using the Pointer mode.

Valid options for both the **LeftButtonOption** and **RightButtonOption** properties are as follows:

- 1 Off
- 2 Pointer
- 3 Highlight
- 4 Redaction
- 5 Text
- 6 Bitmap
- 7 Sticky Note
- 8 Freehand
- 9 Line
- 10 Arrow

1--Off

1--Off disables the Annotation engine and allows the full and normal function of the display window's mouse related events.

- None of the Annotation control's mouse events will occur while in this state.
- Annotation objects will still be displayed and may be created or modified through the programmatic interface.

2--Pointer

2--Pointer allows the selection and editing of existing objects.

- All of the Annotation control's mouse events (**MouseDown**, **MouseUp**, **MouseMove**, **Click**, **DbClick**) can occur while in this state.
- The display window's mouse events still occur normally when the mouse cursor is not over any annotation object.
- Any current annotation object may be selected and moved or sized while in the Pointer mode.
- An object may be selected by clicking on it, but the object will not be made the current object as defined in the **NoteIndex** property. The currently active object may be changed during this process by setting the **NoteIndex** or **NoteName** property in one of the mouse events that occurs when an annotation object is clicked.

3--Highlight

3--Highlight will create a new Highlight object each time the mouse button is depressed. The size and position of the object may be defined by holding down the mouse button and dragging out the area of the highlight.

- The mouse related events (**MouseDown**, **MouseUp**, **MouseMove**, **Click**, **DbClick**) do not occur while in this state.
- When a new object is created, the **NoteCreated** event occurs. This event occurs when the mouse button is initially depressed.

4--Redaction

4--Redaction creates a new Redaction object each time the mouse button is depressed. The size and position of the object may be defined by holding down the mouse button and dragging out the area of the redaction.

- The color and style of the object may be defined before it is created by setting default values. Refer to "Setting Defaults for New Annotations" on page 24 for details on this process.

- The mouse related events **MouseDown**, **MouseUp**, **MouseMove**, **Click**, **DbClick** do not occur while in this state.
- When a new object is created, the **NoteCreated** event occurs. This event occurs when the mouse button is initially depressed.

5--Text

5--Text creates a new Text object when the mouse button is depressed.

- It is not necessary to drag out a region for the object because the font style and the amount of text will be used to calculate the size of the object.
- When the mouse button is depressed, creating the new annotation object, a dialog box is displayed, allowing the user to type in the text for display. When the OK button on the dialog is clicked, the new Text object will be displayed.
- The text to display in the newly created object may be specified in either of two ways:

Set the default value of the **NoteText** property for Text objects before the object is created. Refer to "Setting Defaults for New Annotations" on page 24 for details on this process.

Execute the **ShowTextDialog** method in the **NoteCreated** event. The text that is typed into this dialog will be displayed.

6--Bitmap

6--Bitmap creates a new Bitmap object when the mouse is used to drag out a region for the bitmap.

- The **NoteCreated** event occurs when the new object is created.
- The Windows BMP file that is to be displayed may be specified by entering a file name in the dialog that is opened by executing the **ShowFileDialog** method during the **NoteCreated** event.
- The file for display may be changed by setting the **NoteFileName** property when the **NoteIndex** property is set to select the Bitmap object.

7--Sticky Note

7--Sticky Note creates a new Sticky Note object each time the mouse button is depressed while over image data in the display window that is linked to the Annotation control.

- The **NoteCreated** event occurs when the new object is created.

- The text that is to appear in the Sticky Note object may be specified by setting the **NoteIndex** property to specify the newly created object and then populating the **NoteText** property.
- The file for display may be changed by setting the **NoteFileName** property when the **NoteIndex** property is set to select the Bitmap object.

8--Freehand

8--Freehand creates a new Freehand object when the mouse button is depressed. While the button is being held down, the cursor may be moved to draw any line. When the mouse button is released, the new Freehand object is complete.

9--Line

9--Line creates a new Line object when the mouse button is depressed. While holding down the mouse button, move the cursor to the second end point. When the mouse button is released, the line is drawn and the **NoteCount** property updated.

- The **NoteCreated** event occurs when the new object is created when the button is initially depressed. This event occurs before the **MouseDown** event.
- The line size and color may be specified in either of two ways:
 - 1) The defaults for Line objects may be set to the desired values. Refer to "Setting Defaults for New Annotation" on page 24 for details on this process.
 - 2) After the object is created, **NoteIndex** may be set to specify the new object and the descriptive properties may then be changed for only this object.
- The most commonly used descriptive properties that apply to Line objects are as follows:

NoteThickness

NoteColor

10--Arrow

10--Arrow creates a new Arrow object when the mouse button is depressed. While holding down the mouse button, move the cursor to the second end point. When the mouse button is released, the line is drawn and the **NoteCount** property updated. By default, the arrowhead will be drawn on the second end point of the Arrow object.

- The **NoteCreated** event occurs when the new object is created.

- The line size and color may be specified in either of two ways:
 - 1) The defaults for Arrow objects may be set to the desired values. Refer to "Setting Defaults for New Annotations" on page 24 for details on this process.
 - 2) After the object is created, **NoteIndex** may be set to specify the new object and the descriptive properties may then be changed for only this object.
- The most commonly used descriptive properties that apply to Arrow objects are as follows:

NoteArrowLocation	Specifies at which end of the arrow the arrowhead should appear
NoteThickness	Specifies the thickness of the lines comprising the Arrow
NoteColor	Specifies the color of the Arrow object

Editing Annotations -- Programmatic

Annotation objects can be created and modified entirely through programmatic control. Each type of annotation object (Text, Sticky Note, etc.) is described by a series of properties. These properties can be set to define the desired object, and the object can then be created by executing one of the following methods:

CreateArrow	CreateBitmap
CreateFreehand	CreateHighlight
CreateLayer	CreateLine
CreateRedact	CreateStickyNote
CreateText	

Annotation Object Descriptive Properties

All annotation types are described by a set of basic properties. Each type is further described by a set of properties that is unique to the type. The following pages show these descriptive properties.

- Setting **NoteIndex** to 0 (zero) and **DefaultClass** to a valid value will populate all of the descriptive properties with default values for that class of annotation objects.
- Any changes made to these properties at this time will be reflected when any new annotation object is created. Refer to "Setting Defaults for New Annotations" on page 24 for details on this process.

The object description properties whose defaults may be set as described above are as follows:

Object Description Property	Standard Default Value
NoteArrowAngle	30 (degrees)
NoteArrowLength	100 (pixels)
NoteArrowStyle	0 (Filled)
NoteBackColor	0 (Black)
NoteBottom	0 (image pixels)
NoteColor	0 (Black)
NoteDrawMode	0 (Fit Region to Bitmap)
NoteFileName	NULL
NoteFillPattern	0 (Hollow)
NoteFillStyle	0 (Opaque)
NoteFont.Bold	False
NoteFont.Italic	False
NoteFont.Name	Times New Roman
NoteFont.Size	12
NoteFont.Strikethrough	False
NoteFont.Underline	False
NoteLeft	0 (image pixels)
NoteLineStyle	0 (Solid)
NotePrint	True
NoteRight	0 (image pixels)
NoteSave	True
NoteShape	0 (Rectangle)
NoteText	NULL
NoteThickness	1 (pixel)
NoteTop	0 (image pixels)
NoteVisible	True

Object Description Properties for All Object Classes

The following descriptive properties apply to all annotation types. The default values of these properties may be changed prior to creation of a new annotation object, or they may be modified to edit the current annotation object.

NoteBottom	Image pixel position of the bottom edge of a rectangle that bounds the current annotation object.
NoteClass	Reports the type of the current annotation object according to the following enumerated list: <ul style="list-style-type: none"> -1 Layer 0 Default (Not Used) 1 Reserved (Not Used) 2 Unknown 3 Highlight 4 Redaction 5 Text 6 Bitmap 7 Sticky Note 8 Freehand 9 Line 10 Arrow
NoteColor	Specifies the RGB color of the current annotation object.
NoteLeft	Image pixel position of the left edge of a rectangle that bounds the current annotation object.
NoteName	Reports the name assigned to the current annotation object.
NotePrint	If True (the default), the current annotation object will be included in the printout when the underlying image is printed by the Display control.
NoteRight	Image pixel position of the right edge of a rectangle that bounds the current annotation object.
NoteSelected	Set to True when the object is the currently active object. Multiple objects may be selected by explicitly setting this property to True for those objects.
NoteSave	If True (the default), the current annotation object will be saved when the WriteNotes method is executed or the image page is saved.
NoteTop	Image pixel position of the top edge of a rectangle that bounds the current annotation object.
NoteVisible	If True (the default), the current annotation object will be displayed if the object's LayerVisible and the control's ShowNotes properties are also True.

Creating Arrow Objects

The **CreateArrow** method accepts the image pixel coordinates of the two end points and creates a new Arrow object. For example,

```
Annotel.DisplaySource = TMSDispl
Annotel.CreateArrow  x1, y1, x2, y2
TMSDispl.Refresh
```

The **NoteArrowLocation** property specifies the end on which the arrowhead is drawn. The default is to place the arrowhead at (x2, y2). The options for this enumerated property are as follows:

- 0 Tail End (x2, y2)
- 1 Head End (x1, y1)
- 2 Both

The **NoteArrowAngle** property specifies the angle of the arrowhead. The default is 30 degrees.

The **NoteArrowStyle** specifies whether the arrowhead will be filled or hollow, according to the following enumerated list:

- 0 Filled (Default)
- 1 Hollow

Creating Bitmap Objects

The **CreateBitmap** method displays a BMP file as a new annotation object. This method accepts these parameters:

- x Left edge of the object in image pixels
- y Top edge of the object in image pixels
- File Fully qualified file name of a Windows BMP file

These parameters are used as shown in the sample below:

```
' declare variables
Dim sFileName as String, nNew as Integer

' link Annotation to Display control
Annotel.DisplaySource = TMSDispl.Link

' create the new Bitmap object with a file selected
' through the Common Dialog control as its icon
sFileName = CommonDialog1.ShowOpen
nNew = Annotel.CreateBitmap(100, 1000, sFileName)
TMSDispl.Refresh
```

Creating Freehand Objects

The **CreateFreehand** method creates a new Freehand object by adding the first point. This method accepts parameters that define the image pixel coordinate of the point, as illustrated here:

```
' declare necessary variables
Dim nNew as Integer

' link Annotation to Display control
Annotel.DisplaySource = TMSDispl.Link

' create and select the new object
nNew = Annotel.CreateFreehand(100, 100)
Annotel.NoteIndex = nNew

' add additional points to the object
Annotel.AppendPoint x + 3, y + 1

' refresh the Display to show the new object
TMSDispl.Refresh
```

After a new Freehand object is created, additional points may be added to it using the **AppendPoint** method. When the **AppendPoint** method is executed as shown in the sample above, the **NotePointCount** property is updated.

A point may be removed by specifying it in the **NotePointIndex** property and executing the **DeletePoint** method.

Creating Highlight Objects

The **CreateHighlight** method creates a new Highlight object at the coordinates specified as parameters to the method call. The parameters to this call are the image pixel coordinates of the edges of the highlight. Other characteristics of the object may be specified after the object is created:

```
' link Annotation to Display control
Annotel.DisplaySource = TMSDispl.Link

' create and select the new object
Annotel.NoteIndex = Annotel.CreateHighlight(0, 0, 100,
2000)

' change the color to bright red
Annotel.NoteColor = RGB(225, 0, 0)

' draw object with a diamond fill
Annotel.NoteFillPattern = 7

' draw object as rounded rectangle
Annotel.NoteShape = 1
TMSDispl.Refresh
```

Creating a New Layer

The **CreateLayer** method makes a new layer for additional annotation objects. After the successful creation of the new layer, the **LayerCount** and **LayerIndex** properties are updated to make the new layer the current layer. Until the current layer is changed, all annotation objects subsequently created will be placed on this layer.

The newly created layer will be assigned a name by the control. The name of the layer and its descriptive properties may be modified after the successful execution of the **CreateLayer** method:

```
Annotel.DisplaySource = TMSDispl.Link
Annotel.LayerIndex = Annotel.CreateLayer
Annotel.RenameLayer "Blank Layer"
Annotel.LayerPriority = 0 ' highest priority
Annotel.LayerVisible = True
```

Creating Sticky Note Objects

The **CreateStickyNote** method creates a new Sticky Note object on the current layer, as specified in the **LayerIndex** property. This method call accepts parameters defining the top and left image pixel coordinates of the object and the text that is stored in the object:

```
Dim nNew as Integer
Dim sText as String

' specify the Display that will show the annotations
Annotel.DisplaySource = TMSDispl.Link

' create the new object with the desired text
sText = "This text is displayed in the Sticky Note object"
nNew = Annotel.CreateStickyNote(100, 900, sText)

' select the new object and change some of its
' characteristics, including its icon
Annotel.NoteIndex = nNew
Annotel.NotePicture = LoadPicture("c:\windows\arches.bmp")
Annotel.NotePrint = False

' refresh the Display to show the new object
TMSDispl.Refresh
```

Creating Text Objects

The **CreateText** method adds an arbitrary string of text as a new annotation object on the current layer, as specified in the **LayerIndex** property. If the text extends beyond the edge of the image, the excess data will be lost. The default font and size may be specified before the new annotation object is created:

```
' declare necessary variables and link the
Annotation

' control to a Display control
Dim sText as String
Annotel.DisplaySource = TMSDispl.Link

' set default values for Text objects
Annotel.NoteIndex = -1
Annotel.DefaultClass = 5 ' Text Class

' set defaults for certain font attributes
Annotel.NoteFont.Name = "Book Antiqua"
Annotel.NoteFont.Size = 14
Annotel.NoteFont.Italic = True

' create the new Text object
sText = "This text will be displayed on the image."
Annotel.CreateText 1250, 25, sText

' update the Display control to show the new object
TMSDispl.Refresh
```

Grouping Annotations

The Annotation control offers the option of grouping annotation objects together in layers. Any of the existing layers may be made the active layer. All operation discussed here and in the following sections are performed on the active layer. All of the properties discussed here are populated with the values assigned to the active layer. The following actions may be performed on each layer and will affect all annotation objects on that layer simultaneously:

Make visible or not visible

Move in front of or behind any other layer

Delete entire layer

Creating Layers (Groups) of Annotations

When the Annotation engine initializes, a default layer is created, and new annotation objects can be created on it. All new objects will be placed on this layer.

- Additional annotation layers may be created using the **CreateLayer** method. When this method is executed, a new layer is made and is assigned an index as returned from the method call. A default name is also assigned and is reported in the **LayerName** property.
- At the same time, the **LayerCount** property is updated to reflect the total number of layers currently in memory.
- The **LayerIndex** property may be set to any integer value between 1 and **LayerCount** inclusive, to select an active layer.
- When referencing a particular layer, we recommend that the **LayerName** property be used instead of the **LayerIndex** property because the index assigned to any given layer can change.
- The name of the currently active layer may be changed using the **RenameLayer** method.
- All new annotation objects will be created on the currently active layer.

Properties Describing Layers

The following properties specify characteristics of the active layer and affect the annotation objects on the layer as detailed below:

LayerCount Reports the total number of layers currently defined.

LayerIndex	Specifies the index number of the active layer. If set to any integer between 1 and LayerCount , the specified layer is made the active layer.
LayerName	Specifies the name assigned to the active layer. Setting this property to a valid name will make the named layer the active layer. The name assigned to a given layer may be changed with the RenameLayer method.
LayerPriority	Specifies the position of the layer in the display. A layer with a lesser integer value in this property will be drawn on top of layers with a higher value. Multiple layers may be assigned identical priorities. In this case, the layer ordering is not predictable.
LayerVisible	<p>If False, all annotation objects on this layer will be invisible. If True, each object's NoteVisible property will determine if the object can be seen.</p> <p>Applies only if the ShowNotes property is True. If ShowNotes is False, all annotation objects will be invisible.</p>

Deleting an Annotation Layer

When an annotation layer is deleted, all annotation objects on that layer are also deleted. All of the deleted objects are permanently destroyed and are not recoverable unless previously saved.

A layer is deleted using the **DeleteLayer** method after selecting the layer to delete by setting the **LayerIndex** or **LayerName** property. This method requires no parameters and returns a 0 on success and a negative integer error code on failure. The display will not show the changes until it is refreshed:

```

Annotel.DisplaySource = TMSDispl.Link
Annotel.DeleteLayer
TMSDispl.Refresh

```

When a layer and the annotation objects on that layer are deleted, the remaining annotation objects will be assigned new **NoteIndex** values to fill in any gaps created by the deletion. When a layer is deleted, the **NoteCount** property is also updated to reflect the new total of annotation objects.

Because the **NoteIndex** assigned to a particular object will change, any tracking of individual annotation objects should be done through the **NoteName** property, which does not change, rather than through the **NoteIndex** property which can be changed by many different actions.

Removing Annotation Objects

Three techniques are available for removing annotations:

- 1) A single annotation object, such as a Redaction region or a Freehand object, can be deleted using the **DeleteNote** method.
- 2) All annotation objects on a single layer may be deleted by removing that layer using the **DeleteLayer** method.
- 3) All annotations currently in memory can be removed by executing the **ClearNotes** method.

Deleting One Annotation Object

The annotation object specified in the **NoteIndex** property can be deleted by executing the **DeleteNote** method:

```
Annotel.DisplaySource = TMSDispl.Link  
Annotel.NoteName = "Annote 12"  
Annotel.DeleteNote  
TMSDispl.Refresh
```

When an annotation object is deleted, all other annotation objects whose **NoteIndex** value is greater than the deleted objects value will be assigned new **NoteIndex** values. When the object is deleted, the **NoteCount** property is also updated to reflect the new total of annotation objects.

For this reason, any tracking of individual annotation objects should be done through the **NoteName** property, which does not change, rather than through the **NoteIndex** property which can be changed by many different actions.

Deleting One Annotation Layer

Any given layer may be deleted from memory by executing the **DeleteLayer** method after specifying the layer to delete in the **LayerIndex** property:

```
Annotel.DisplaySource = TMSDispl.Link  
Annotel.LayerName = "Layer 2"  
Annotel.DeleteLayer  
TMSDispl.Refresh
```

When a layer (and the annotation objects on that layer) are deleted, the remaining annotation objects will be assigned new **NoteIndex** values to fill in any gaps created by the deletion. When a layer is deleted, the **NoteCount** property is also updated to reflect the new total of annotation objects.

Because of the transitory attachment of a particular **NoteIndex** to any given annotation object, any tracking of individual annotation objects should be done through the **NoteName** property, which does not change, rather than through the **NoteIndex** property which can be changed by many different actions.

Deleting All Annotation Objects

To remove all annotation objects, execute the **ClearNotes** method:

```
Annotel.DisplaySource = TMSDispl.Link  
Annotel.ClearNotes  
TMSDispl.Refresh
```

This call will remove all current annotations. The objects will be removed from memory and will be permanently lost unless previously saved. The **NoteCount** property will be set to 0 (zero) and all descriptive properties will be set to defaults.

Viewing and Printing Annotations

Multiple annotation objects can be created on a single image or stored in a single location. In many cases, only a portion of these individual objects should be displayed to the user at any one time. For this reason, the Annotation control can control the visibility of annotation objects either individually or in groups.

A group of annotations is referred to as a layer. By default, all annotations will be created on the same layer. Each annotation object can be made independently visible or invisible. Multiple layers can be created and populated with annotation objects, and each of these layers can be made visible or invisible as a group.

Displaying and Hiding Annotations

When a new annotation object is created or an existing object is modified, the object is held in memory by the Annotation control. The new object will not be displayed until the display window is refreshed. To ensure that all new objects are immediately displayed, execute the display window **Refresh** method after each annotation object is created or modified.

When an annotation object is being held in memory (whether it was loaded from file or just created on the image) it can be made visible or invisible to the user. The visibility of annotation objects is controlled by three inter-related properties:

ShowNotes	If False, all annotation objects will be invisible. If True, the annotation objects may be visible, depending upon the values in the following two properties.
LayerVisible	If False, all annotation objects that are drawn on the current layer will be invisible. If True, each individual annotation object on the current layer will be visible or invisible depending upon the value of the NoteVisible property.
NoteVisible	If False, the current annotation object will be invisible. If True, the current annotation object will be visible.

Note: The three properties above are applied in the priority listed. For example, if the **ShowNotes** property is False, no annotation objects will be visible, even if the **LayerVisible** and **NoteVisible** properties are True. Likewise, if the **LayerVisible** property is False, all annotation objects on that layer are invisible, even if the **NoteVisible** property is True for each of the individual objects.

Ordering the Display of Annotations

As annotation objects are created, overlapping objects will be drawn on the display and printed with the newest objects drawn on top of older objects. The display order of the objects is set through the **NotePriority** property.

- The value of the **NotePriority** property reports the display priority assigned to the current object. Refer to the 'NoteIndex Property' on page 78 or the 'NoteName Property' on page 81 for details on selecting the current object.
- Any object assigned a lesser (closer to zero) **NotePriority** value will be displayed on top another object with a greater **NotePriority**.
- At their creation, all objects are assigned a **NotePriority** of 0 (zero), the highest priority. Because **NotePriority** is unique on each layer, all other objects on that layer may be assigned new **NotePriority** values to avoid conflicts.
- The new object will therefore be displayed on top of all other objects on that layer.
- The **BringToFront** method moves the currently selected annotation object to the top of the display stack. This is equivalent to setting the object's **NotePriority** to 0 (zero).
- The **SendToBack** method moves the current annotation object to the bottom of the display stack. This is equivalent to setting the object's **NotePriority** to **NoteCount** + 1.

- Changing the **NotePriority** of an object can change the **NoteIndex** values assigned to all objects. It is therefore necessary to refer to and select annotation objects through the **NoteName** property rather than through the **NoteIndex** property when selecting particular objects. The **NoteIndex** property is designed primarily for looping through all objects in conjunction with the **NoteCount** value.

Printing Annotations with an Image

By default, all currently visible annotation objects will be printed when the image on which they are displayed is printed. This default behavior can be modified through two properties:

PrintNotes

NotePrint

PrintNotes is a Boolean property that enables and disables the printing of all annotation objects. This property is analogous to the **ShowNotes** property which controls the display of annotation objects in a similar fashion.

- If **PrintNotes** is True, all objects can be printed, depending upon the value of each object's **NotePrint** property.
- If **PrintNotes** is False, none of the annotation objects will be printed on the image.

NotePrint is a Boolean property that enables and disables the printing of a single annotation object. This property is analogous to the **NoteVisible** property which controls the visibility of an individual object in a similar fashion.

The value of this property for any given object may be read or set after selecting the object by setting the **NoteIndex** or **NoteName** property.

- If **NotePrint** is True, the current object will be printed as long as **PrintNotes** is also True.
- If **NotePrint** is False, the current object will not be printed.

When annotations are printed on an image, the objects will be dithered or colored as necessary to allow the proper visibility of the underlying image data. For example, a highlighted region will be printed as a dithered gray area so that any image text underneath it can be read. In contrast, a redaction region will be printed such that the underlying image is completely obscured.

Saving and Loading Annotations

A group of annotation objects that are held in memory may be saved in these different formats:

- In the header record of a TIFF file
Annotations saved in this manner may be written to any page in a multiple page TIFF. Each page can contain a separate set of annotations.
- A separate annotation file
Annotations saved in a separate file are not associated with any particular image file. The application must maintain the relationship between the annotation file and the image file that matches it.
- Imprinted on the image data **Not implemented**
Rather than save annotations in an editable format, it may be necessary or desirable under certain circumstances to permanently meld the annotations with the image data. In this case, all of the annotation objects may be imprinted onto the underlying image data and saved with it when the image is written to a file.

Saving and Loading Annotations in TIFF Files

The Annotation control offers the option of saving annotation data directly to a TIFF file's header, so that the information is available whenever that TIFF file is loaded into an ImageBASIC Display control. The file access required for reading and writing to the TIFF header is performed automatically once the **AnnoteSource** property is set to a valid value.

Loading Annotations from a TIFF Header

Whenever a File control loads an TIFF file that contains supported annotations in the header, those annotations are loaded into memory.

- The annotation data is made a part of the sideband data of the image structure. This entire image structure (the image data plus its sideband data) is available to any ImageBASIC control that is linked to the File control.
- When a Display control links to this File control, any Annotation control linked to that Display has access to the annotation sideband data for each image that is displayed.
- Changes to the annotations for the displayed image will be written to the sideband data for the image page. When that image page is saved

through an ImageBASIC File control, the annotation data is written to the TIFF header.

Therefore, simply linking an Annotation control to a Display control automatically reads and displays all supported annotations in the image file header:

```
' annotations are shown in the Display control
Annotel.DisplaySource = TMSDispl.Link

' images and associated data are loaded by the File
' control
TMSDispl.ImageDataSource = TMSFile1.Link
```

Saving Annotations to a TIFF Header

Annotations will be saved to a TIFF file's header when the image on which they are displayed is saved if the following are True:

- 1) The Annotation control maintains its link to the Display control; i.e., the **DisplaySource** property of the Annotation control specifies the Display control.
- 2) The File control's **ImageDataSource** must specify the same Display control as the Annotation control's **DisplaySource**
- 3) The File control saves the image from the Display control.

Note: The File control must save the image from the Display control. Refer to a File control user's guide for details on the source of image data for writing to file.

The following code snippet will load an image, create annotations on the image, and then save the image to a new file with the annotations in the file's header:

```
' link the Display control to a File control to
' display an image
TMSDispl.ImageDataSource = TMSFile1.Link

' link the Annotation control to the Display
control;
' this Display control will show all annotations
made
' by the Annotation control
Annotel.DisplaySource = TMSDispl.Link
```

< Continued >

```

' load an image; for this example, assume a multi-
' page TIFF; the first page of the file will be
' displayed
TMSFile1.LoadFile "c:\images\multpage.tif"

' load and display the second page of the file
TMSFile1.PageIndex = 2

' add one or more annotation objects to the image;
' for this example, add a new Text object and
refresh
' the Display control
Annotel.CreateText 100, 100, "This is a Text annotation"
TMSDispl.Refresh

' link the File control back to the Display control
' so that the modified page can be put back into the
' original document; the annotation sideband data
' will be supplied to the File control along with
the
' image data
TMSFile1.ImageDataSource = TMSDispl.Link

' replace the current page of the File control's
' document with the new page from the Display
control
TMSFile1.ReplacePage

' save the new (changed) document; the new
annotation
' object will be written to the file header so that
' it is available for future display or
modification;
' Note: Only TIFF files support this type of saving
TMSFile1.OutputFileFormat = 21      ' TIFF Group 3 2-D

' add the document pages to the existing output file
TMSFile1.OutputFileAppend = True

' save all pages of the document along with any
' annotations on each page
TMSFile1.SaveDocument "c:\newfile.tif"

```


Saving and Loading Annotation Files

Annotations may be saved to any of the supported annotation file formats. As of this writing, the Wang annotation specification is the most widely accepted standard for annotation storage and retrieval, and is therefore the default for writing of annotation files. To maximize compatibility with existing systems and to simplify for future expansion, the Annotation control fully supports this standard.

When annotation objects are saved, either to a separate file or to a TIFF header record, the **NoteSave** property for each annotation object will determine if the object is saved to file. All annotation objects that are currently held in memory and whose **NoteSave** property is True will be saved, even if they are not visible.

Saving Annotation Files

For easy integration with existing systems or other integration efforts, the ImageBASIC Annotation control is designed to read and write from Wang-compatible annotation files.

The annotation file is not inherently associated with any image file. Maintaining the link between the annotation file and an image file, including the correct page in a multiple page image file, is left to the developer's discretion.

Annotation files are written using the **WriteNotes** method. When executed, this method will create the specified annotation file in the specified format. If the file already exists, it will be overwritten. The syntax and parameters to this method are as follows:

```
Annotation1.WriteNotes  filename, format
```

filename is a string parameter of the name of the file with a relative or absolute path

format is an enumerated (integer) parameter specifying the file type to write, based on the following list:

- | | |
|--------------|--|
| 1--DHS | Not yet implemented |
| 2--Wang | Default value; format used by earlier ImageBASIC 3.x revisions |
| 3--Pixel | Not implemented |
| 4--Watermark | Not implemented |

Note, only the DHS annotation specification allows for saving any information in the **NoteData** property.

Example: Saving an Annotation File

The file containing the annotations might be associated with the image file by giving the annotation file the same name as the image file name, but with a different extension.

The following code will create a file with the same name as the currently displayed image file name, but with the extension of ANT. The annotation markings that were created by the control named `Annotel` and are currently active are then saved in this file. For the purposes of this example, the assumption is made that this code is attached to a command button named *SaveAnnote*:

```
' declare necessary variables
Dim sName as String

' link the ImageBASIC controls to data sources
TMSDispl.ImageDataSource = TMSFile1.Link
Annotel.DisplaySource = TMSDispl.Link

' read the name of the currently loaded image file
sName = TMSFile1.InputFileName

' create annotation file name from image file name
sName = Left$(sName, InStr(sName, "."))
sName = sName + ".ant"

' save annotations in Wang format to the newly
' created file name
Annotel.WriteNotes sName, 2
```

Loading Annotation Files

Any annotation file that is compatible with one of the supported specifications for annotation files may be read by the ImageBASIC Annotation control using the **ReadNotes** method. All objects in the file are loaded into memory and are available for display. The format of the file is specified as an enumerated argument to the method call:

```
Annotel.ReadNotes filename, format
```

filename is a string parameter of the name of the file with a relative or absolute path

format is an enumerated (integer) parameter specifying the file type that is being read, based on the following list:

0--Auto Detect	Default value
1--DHS	Not yet implemented
2--Wang	Wang specification annotation files as used in ImageBASIC 3.x revisions
3--Pixel	PixTools specification annotation files as used in ImageBASIC 2.2
4--Watermark	Not yet implemented

Example: Loading an Annotation File

Any separate annotation file that matches the specifications of the supported format can be read into memory using the **ReadNotes** method, as shown below:

```
' link Annotation to a Display control
Annotel.DisplaySource = TMSDispl.Link

' read the annotation objects from a file
Annotel.ReadNotes "c:\notel.ant" , 0 ' Autodetect type

' refresh the display window to show the new
objects
TMSDispl.Refresh
```

The fully qualified file name for any annotation file may be specified as the parameter to this method. Refer to these related topics for additional information:

"Saving Annotation Files" on page 45

"Saving and Loading Annotations in TIFF Files" on page 42

Imprinting Annotations into an Image

Imprinting is not yet implemented in the Annotation control.

When a set of annotations should be permanently saved with an image, the annotations may be imprinted onto the image itself. This technique merges the annotations with the image data and allows them to be saved as an integral part of the image. All currently visible annotations will be imprinted onto the image data when this method is executed.

When imprinted, annotations are converted to the color format of the underlying image. If necessary, the annotations will be dithered or otherwise modified to maintain the visibility of underlying image data.

In the following example, an existing ImageBASIC annotation file is loaded and the annotations contained in it are merged into the underlying image:

```

        ' link all ImageBASIC controls to data sources
Annotel.DisplaySource = TMSDispl.Link
TMSDispl.ImageDataSource = TMSFile1.Link
TMSFile1.ImageDataSource = TMSDispl.Link

        ' load an image file
TMSFile1.LoadFile "c:\images\image001.tif"

        ' load an annotation file
Annotel.ReadNotes "c:\annote\notes002.ant", 2

        ' refresh the display with new data
TMSDispl.Refresh

        ' imprint the annotations onto the image
Annotel.Imprint

        ' save the image with annotations to a new file
TMSFile1.Save "c:\images\image002.tif"

```

Note: After annotations are imprinted onto an image, the annotation objects cannot be modified -- they are fully converted to image data. The image must be saved to maintain the newly merged annotations.

Chapter 3 : Reference

Reference List

The following pages contain detailed descriptions of all of the properties, methods and events of this control.

- This chapter should answer any questions concerning any individual feature of the control.
- The chapter titled 'Chapter 2 : Using the Annotation Control' on page 7 contains more thorough discussions of the interaction of these features when performing data manipulation.
- If you are learning this control, please begin with the introduction to this control found in 'Chapter 1 : Getting Started' on page 1.

AboutBox Method

Definition:	Displays a message box containing the name of the control, a copyright message and an OK button. Pressing the button will unload the message box.
Parameters:	None
Syntax:	<code>Annotel>AboutBox</code>
Return Value:	None
Comments:	This message box provides information about the Annotation control.

Active Property

Definition:	<p>If set to True at design time, the control will fully initialize and verify licensing immediately upon initialization of the runtime application.</p> <p>If set to False at design time, full initialization of the control will be delayed at initialization of the runtime application. In this case, this property must be explicitly set to True at runtime before the control is used.</p>
Data Type:	Boolean
Design Access:	Read/Write
Runtime Access:	Read/Write (see limits below)
See Also:	"Licensing Configuration and Verification" on page 4

Comments: If this property is set to True (the default) at design time, the control is fully initialized and licensing is verified immediately upon initialization of the application at runtime. The technology libraries are loaded and the control is ready to be used.

If this property is set to False at design time, the control will only partially initialize when the application loads at runtime. By delaying these two actions, the application should be able to load more quickly:

- 1) The technology libraries for the control will not be loaded.
- 2) The licensing server will not verify an available token for the control.

If the control initializes with **Active** set to False, this property must be explicitly set to True by the application. Until **Active** is set to True, the control will ignore all instructions to it.

If the control fails to find a license token, the Active property will be automatically set to False. The application can check this value on Form Load to determine if each control is licensed and can be used.

AppendPoint Method

Definition: Adds a new point to the Freehand object that is specified in the **NoteIndex** property.

Parameters: x Horizontal image pixel coordinate of point
y Vertical image pixel coordinate of point

Syntax: `Annotel.AppendPoint x, y`

Return Value: NotePointCount on success
-1 on failure

Data Type: Long

See Also: CreateFreehand Method, NotePointCountProperty

Comments: The **NoteIndex** or **NoteName** property must be set to specify a Freehand object before this method is executed. After successful completion, the following properties will be updated:

- NotePointCount
- NoteBottom
- NoteLeft
- NoteRight
- NoteTop

BringToFront Method

Definition:	Changes the display priority of the current annotation object so that it is displayed on top of all other objects on that layer.
Parameters:	None
Syntax:	<code>Annotel.BringToFront</code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	SendToBackMethod, NotePriority Property
Comments:	Executing this method will alter the current object's NotePriority to 0 (zero) and NoteIndex to 1 (one) so that the object is on the top of the display stack. This will usually cause all other objects' NotePriority and NoteIndex values to change as well.

ClearNotes Method

Definition:	Removes all annotation objects currently in memory.
Parameters:	None
Syntax:	<code>Annotel.ClearNotes</code>
Return Value:	0 on success -1 on failure
Data Type:	Integer
See Also:	DeleteNoteMethod
Comments:	After successful completion NoteIndex will be set to 0 (zero) and all descriptive properties will be reset to their default values.

Click Event

Definition:	Occurs when the Annotation control is in its Pointer mode (see the LeftButtonOption property), and the user clicks on an existing annotation object.
Parameters:	LayerName Name of the layer containing the clicked object LayerIndex Index of the layer containing the clicked object NoteName Name of the clicked object NoteIndex Index of the clicked object
See Also:	DbClick Event
Comments:	While in the Pointer mode, if the user clicks on the displayed image outside the boundaries of an of the displayed annotation

objects, the display window's **Click** event will occur, but not the Annotation control's **Click** event.

CreateArrow Method

Definition: Creates a new Arrow annotation object at the coordinates specified in the method call. All parameters are integers.

Parameters:

x1	X image pixel coordinate of tail of arrow
y1	Y image pixel coordinate of tail of arrow
x2	X image pixel coordinate of head of arrow
y2	Y image pixel coordinate of head of arrow

Syntax: `Annotel.CreateArrow x1, y1, x2, y2`

Return Value: Index of object on success
-1 on failure

Data Type: Integer

See Also: "Editing Annotations -- Programmatic" on page 29

Comments: When a new Arrow object is created by the execution of this method, the new object is held in memory but is not automatically displayed. At this time, any of the descriptive properties of the new object may be changed. As shown below, the new object may be selected and modified before the display window is redrawn:

```
Dim nNew as Integer
Annotel.DisplaySource = TMSDispl
nNew = Annotel.CreateArrow 55, 10, 500, 900
Annotel.NoteIndex = nNew
Annotel.NoteColor = RGB(0, 0, 220)
Annotel.NoteArrowAngle = 45
TMSDispl.Refresh
```

CreateBitmap Method

Definition: Creates a new Bitmap annotation object at the coordinates specified in the method call.

Parameters:

Left	Integer image pixel coordinate of left of bitmap
Top	Integer image pixel coordinate of top of bitmap
FileName	File name of BMP to insert

Syntax: `Annotel.CreateBitmap left, top, filename`

Return Value: Index of object on success
-1 on failure

Data Type: Integer

See Also: NoteDrawMode Property, Editing Annotations -- Programmatic" on page 29

Comments: Any valid uncompressed Windows BMP file may be loaded as an annotation using this method. By modifying the **NoteDrawMode** property, the file may be scaled to fit any arbitrary region or may be displayed at full size. In the following example, the BMP is scaled to fill the current Working Region:

```
Dim nTop as Integer, nLeft as Integer
Dim nBottom as Integer, nRight as Integer
Annotel.DisplaySource = TMSDispl
nTop = TMSDispl.RegTop
nLeft = TMSDispl.RegLeft
nBottom = TMSDispl.RegBottom
nRight = TMSDispl.RegRight
Annotel.CreateBitmap nLeft, nTop, "c:\1.bmp"
Annotel.NoteBottom = nBottom
Annotel.NoteRight = nRight
Annotel.NoteDrawMode = 1 ' Fit Bitmap to Region
TMSDispl.Refresh
```

CreateFreehand Method

Definition: Creates a new Freehand annotation object starting at the point specified in the method call. Additional points may be added to the object using the **AppendPoint** method.

Parameters: x Integer horizontal image coordinate of point
y Integer vertical image pixel coordinate of point

Syntax: Annotel.CreateFreehand x, y

Return Value: Index of object on success
-1 on failure

Data Type: Integer

See Also: AppendPoint Method, Editing Annotations -- Programmatic" on page 29

Comments: When this method is executed, the new Freehand object is created and held in memory. Additional points may be added to the object using the **AppendPoint** method. As the object is modified, the **NoteTop**, **NoteLeft**, **NoteBottom** and **NoteRight** properties will be adjusted to reflect the object.

Note: When the new object is created or modified, the display does not automatically display it. The new object will be shown when the window is next drawn -- the application can force the drawing by executing the Display control's **Refresh** method.

CreateHighlight Method

Definition: Creates a new Highlight annotation object at the coordinates specified in the method call.

Parameters:

Top	Top edge of new object in image pixels; integer
Left	Left edge of new object in image pixels; integer
Right	Right edge of object in image pixels; integer
Bottom	Bottom edge of object in image pixels; integer

Syntax: `Annotel.CreateHighlight top, left, right, btn`

Return Value: Index of object on success
-1 on failure

Data Type: Long

See Also: CreateRedaction Method, Editing Annotations -- "Programmatic" on page 29

Comments: When this method is executed, a new Highlight object is created but is not displayed until the display window is redrawn. Immediately after creation, the object may be modified before it is displayed as illustrated in the following code segment:

```
Annotel.DisplaySource = TMSDispl  
nRet = Annotel.CreateHighlight(50, 50, 250,  
    250)  
  
Annotel.NoteIndex = nRet  
Annotel.NoteColor = RGB(100, 100, 0)  
Annotel.NoteFillPattern = 6 ' Fill with grid  
Annotel.NoteShape = 1 ' Rounded Rectangle  
TMSDispl.Refresh
```

CreateLayer Method

Definition: Creates a new layer on which new annotation objects may be grouped.

Parameters: None

Syntax: `Annotel.CreateLayer`

Return Value: Layer index on success
-1 on failure

Data Type: Long

See Also: LayerIndex Property, LayerName Property, RenameLayer Method

Comments: When this method is used to create a new layer, the layer is assigned an index number and a name. After a layer is created, it may be made the active layer by setting the **LayerIndex** property to the return value from this method.

The following properties report the characteristics of the currently loaded layers. An active layer may be selected by setting either the **LayerIndex** or the **LayerName** property to a valid value.

- LayerCount
- LayerIndex
- LayerName
- LayerPriority
- LayerVisible

LayerCount is read-only and reports the total number of layers currently loaded.

LayerIndex specifies the current layer. An active layer may be selected by setting either **LayerIndex** or **LayerName**.

LayerName is automatically populated when the layer is created and may be modified only by executing the **RenameLayer** method.

LayerPriority specifies the order in which multiple layers are stacked. Layers with a lower **LayerPriority** are drawn closer to the top.

LayerVisible specifies whether or not the annotation objects on this layer will be displayed. If True, each object's **NoteVisible** property will be verified to determine if the object is visible. If False, all objects on this layer will not be visible.

Refer to "Grouping Annotations" on page 36 for details on the implementation of layers in the Annotation control.

CreateLine Method

Definition: Creates a new Line annotation object between the two points specified in the method call.

Parameters:

x1	Integer X image pixel coordinate of tail of line
y1	Integer Y image pixel coordinate of tail of line
x2	Integer X image pixel coordinate of head of line
y2	Integer Y image pixel coordinate of head of line

Syntax: Annotel.CreateLine x1, y1, x2, y2

Return Value: Index of object on success

Data Type: Long

See Also: "Editing Annotations -- Programmatic" on page 29, **LayerIndex** Property

Comments: When this method is successfully executed, a new **Line** object is created on the currently active layer as specified in the **LayerIndex** property, and the **NoteCount** property is updated to reflect the new count.

The display control linked to this Annotation control will not be automatically updated but may be explicitly refreshed. The newly created **Line** may be modified before it is displayed as shown in the following segment of code:

```
Dim nNew as Integer
Annotel.DisplaySource = TMSDispl
nNew = Annotel.CreateLine (50, 50, 1000, 1000)
Annotel.NoteIndex = nNew
Annotel.NoteThickness = 3
Annotel.NoteColor = &HCF00A1
Annotel.RenameNote "Line #2003"
Annotel.NoteSave = False
Annotel.NotePrint = True
Annotel.NoteVisible = True
TMSDispl.Refresh
```

CreateRedaction Method

Definition: Creates a new *Redaction* annotation object at the coordinates specified in the method call.

Parameters:

Top	Top edge of new object in image pixels; integer
Left	Left edge of new object in image pixels; integer
Right	Right edge of object in image pixels; integer
Bottom	Bottom edge of object in image pixels; integer

Syntax: `Annotel.CreateRedaction top, left, right, bttm`

Return Value: Index of object on success
-1 on failure

Data Type: Long

See Also: **CreateHighlight Method**, **NoteCount Property**, "Editing Annotations -- Programmatic" on page 29

Comments: When this method is successfully executed, a new **Redaction** object is created on the currently active layer, as specified in the **LayerIndex** property, and the **NoteCount** property is updated to reflect the new count.

The display control linked to this Annotation control will not be automatically updated but may be explicitly refreshed. The newly created Redaction may be modified before it is displayed as shown in the following segment of code:

```

Dim nNew as Integer
Annotel.DisplaySource = TMSDispl
nNew = Annotel.CreateRedaction (5, 5, 100,
    100)

Annotel.NoteIndex = nNew
Annotel.NoteColor = &H505050 ' dark gray
Annotel.NoteShape = 2 ' Rounded Rectangle
Annotel.NoteSave = True
Annotel.NotePrint = False
Annotel.NoteVisible = True
TMSDispl.Refresh

```

CreateStickyNote Method

- Definition:** Creates a new Sticky Note annotation object.
- Parameters:**
- | | |
|----------|--|
| x | Image coordinate of left edge of object; integer |
| y | Image pixel coordinate of top of object; integer |
| Text | Text of the annotation object |
| Caption | Caption for the object |
| FileName | File name of BMP to display as object |
- Syntax:** Annotel.CreateStickyNote *x, y, text, cpt, file*
- Return Value:** Index of object on success
-1 on failure
- Data Type:** Long
- See Also:** CreateBitmap Method, 'Sticky Note Annotation Class' on page 20, 'Editing Annotations -- Programmatic' on page 29
- Comments:** When this method is successfully executed, a new Sticky Note object is created on the currently active layer, as specified in the **LayerIndex** property, and the **NoteCount** property is updated to reflect the new count.

The display control linked to this Annotation control will not be automatically updated but may be explicitly refreshed. The newly created Sticky Note may be modified before it is displayed as shown in the following segment of code:

```

Dim sText as String, nNew as Integer
Annotel.DisplaySource = TMSDispl
sText = "This will be the text in the note"
nNew = Annotel.CreateStickyNote 0, 0, sText
Annotel.NoteIndex = nNew
Annotel.NoteFont.Name = "Arial Narrow"

```

```

Annotel.NoteFont.Size = 12
Annotel.NoteSave = True
Annotel.NotePrint = False
Annotel.NoteVisible = True
TMSDispl.Refresh

```

When a Sticky Note object is displayed, it may be opened and read by double clicking on the object while the mouse pointer is in Pointer mode. Refer to the **LeftButtonOption** property for details on setting these modes.

CreateText Method

- Definition:** Creates a new Text annotation object.
- Parameters:**
- x* Image coordinate of left edge of object; integer
 - y* Image coordinate of top of object; integer
 - text* Text string to be displayed
- Syntax:** `Annotel.CreateText x, y, text`
- Return Value:** Index of object on success
-1 on failure
- Data Type:** Long
- See Also:** NoteFont Property, NoteIndex Property, "Editing Annotations -- Programmatic" on page 29
- Comments:** When this method is successfully executed, a new Text object is created on the currently active layer, as specified in the **LayerIndex** property, and the **NoteCount** property is updated to reflect the new total of annotation objects.

The display control linked to this Annotation control will not be automatically updated but may be explicitly refreshed. The newly created Text object may be modified before it is displayed as shown in the following segment of code:

```

Dim sMessage as String, nNew as Integer
Annotel.DisplaySource = TMSDispl
sMessage = "Display Me!"
nNew = Annotel.CreateText (10, 20, sMessage)
Annotel.NoteIndex = nNew
Annotel.NoteColor = RGB(200, 0, 0)
Annotel.NoteFont.Name = "Arial"
Annotel.NoteFont.Size = 14
Annotel.NoteSave = True
Annotel.NotePrint = True
Annotel.NoteVisible = True
TMSDispl.Refresh

```

DbClick Event

Definition:	Occurs when in Pointer mode, and the user double clicks on an existing annotation object.		
Parameters:	LayerName	Name of the layer containing the clicked object	
	LayerIndex	Index of the layer containing the clicked object	
	NoteName	Name of the clicked object	
	NoteIndex	Index of the clicked object	
See Also:	LeftButtonOption Property, RightButtonOption Property		

DefaultClass Property

Definition:	When changing default values for the creation of new annotation objects, this property specifies the class whose default values are being set.										
Data Type:	Enumerated (Integer)										
Design Access:	Read/Write										
Runtime Access:	Read/Write										
See Also:	NoteIndex Property, 'Setting Defaults for New Annotations' on page 24										
Comments:	<p>Valid options for this property are as follows:</p> <ul style="list-style-type: none">1 Highlight2 Redaction3 Text4 Bitmap5 Sticky Note6 Line7 Arrow8 Freehand <p>Default values for many of the descriptive properties may be set as shown below:</p> <ul style="list-style-type: none">1) Set DefaultClass to specify the object class2) Set NoteIndex to 0 (zero), indicating property defaults3) Set the desired property values <p>The following properties may be set:</p> <table><tr><td>NoteColor</td><td>NoteFillPattern</td></tr><tr><td>NoteDrawMode</td><td>NoteFont</td></tr><tr><td>NoteFillStyle</td><td>NoteShape</td></tr><tr><td>NoteLineStyle</td><td>NoteThickness</td></tr><tr><td>NoteText</td><td></td></tr></table>	NoteColor	NoteFillPattern	NoteDrawMode	NoteFont	NoteFillStyle	NoteShape	NoteLineStyle	NoteThickness	NoteText	
NoteColor	NoteFillPattern										
NoteDrawMode	NoteFont										
NoteFillStyle	NoteShape										
NoteLineStyle	NoteThickness										
NoteText											

DeleteLayer Method

Definition:	Deletes the layer specified in LayerIndex . This also deletes all annotation objects on that layer.
Parameters:	None
Syntax:	<code>Annotel.DeleteLayer</code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	LayerIndex Property, ClearNotes Method
Comments:	<p>When a layer is deleted by the execution of this method, all annotation objects on that layer are permanently destroyed. At this time, the LayerCount and NoteCount properties are updated. The index numbers assigned to any remaining layers and annotation objects will be modified, if necessary, so that all objects are sequentially numbered.</p> <p>The display control to which this annotation controls linked must be refreshed to display the remaining annotations without the deleted layer.</p> <p>If a layer of annotations is to be temporarily hidden, the LayerVisible property may be set to False instead of using the DeleteLayer to destroy the layer.</p>

DeleteNote Method

Definition:	Deletes the single annotation object specified in the NoteIndex or NoteName property.
Parameters:	None
Syntax:	<code>Annotel.DeleteNote</code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	DeleteLayer Method, ClearNotes Method, LeftButtonOption Property, NoteIndex Property
Comments:	<p>Any single annotation object may be selected by setting the NoteIndex or NoteName property to specify the object. An object may be selected by clicking if the LeftButtonOption or RightButtonOption is set to 2--<i>Pointer</i> and the NoteIndex property is set in the Click event.</p> <p>When an annotation object is deleted by the successful execution of this method, the NoteIndex property will be set to 0 (zero).</p>

DeletePoint Method

Definition:	Deletes the current point of a Freehand class object.
Parameters:	None
Syntax:	<code>Annotel.DeletePoint</code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	NotePointIndex Property, NoteClass Property, AppendPoint Method
Comments:	<p>Any one point of a Freehand object may be deleted using this method. To select the point to delete, set the NotePointIndex property to any valid value.</p> <p>After this method is successfully executed, the NotePointCount property is updated. The displayed object will not be changed until the Display control is refreshed.</p>

DisplaySource Property

Definition:	Specifies the ImageBASIC display control that this Annotation control will use to display its objects.
Data Type:	String
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	"Displaying Annotations on an Image" on page 2
Comments:	<p>This property may be set to any ImageBASIC display control, including one on a different Form. If the display control is on a different Form, the Form must be specified with the control name as shown here:</p> <pre>Annotel.DisplaySource = frmViewer!TMSDisp1</pre> <p>All annotation objects will be drawn on this Display control. Each time that any annotation object is modified through code (rather than through the graphical interface described under "Editing Annotations -- Graphical" on page 25), the display control must be explicitly refreshed to draw the changes.</p>

Error Event

Definition:	Occurs for each error internal to the control.	
Parameters:	Number	A long error code that identifies the error
	Description	Descriptive string of the error
	SCode	Equal to the Visual Basic runtime error code which is the error number reported in the <i>Number</i> parameter plus the Visual Basic constant vbObjectError.
	Source	Descriptive string of the source of the error
	HelpFile	Suggested help file name that should have a detailed explanation of the error
	HelpContext	Context ID of the appropriate topic in the help file named above
	CancelDisplay	If set to True during this event, the standard error dialog will not be displayed
Comments:	Any time an error occurs inside the Annotation control, the Error event is triggered.	

LayerCount Property

Definition:	Reports the total number of annotation layers currently defined in memory.
Data Type:	Long
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	LayerIndex Property
Comments:	<p>The value of this property will be updated each time any of the following actions are performed:</p> <ul style="list-style-type: none">A layer is removed using the DeleteLayer methodA layer is added using the CreateLayer methodAnnotations are loaded from file using the ReadNotes method or through the AnnoteSource property <p>The LayerIndex property may be set to any integer value between 1 (one) and LayerCount inclusive.</p>

LayerIndex Property

Definition:	When set to a valid value, selects a single layer from the available layers and makes it the active layer.
Data Type:	Long
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	LayerCount Property, LayerName Property
Comments:	<p>This property may be set to any integer value between 1 (one) and LayerCount. Setting this property to any valid setting will populate the following properties defining this layer:</p> <ul style="list-style-type: none">LayerNameLayerPriorityLayerVisible <p>All newly created annotation objects will be placed on the layer selected in the LayerIndex property.</p> <p>Note: A layer may be selected as the active layer by setting either the LayerIndex or the LayerName property. It is recommended that any application that tracks or maintains records of layer contents use the LayerName property rather than the LayerIndex property to identify the layer. The LayerIndex of any given layer is subject to change when other layers are created or deleted.</p>

LayerName Property

Definition:	Specifies the name assigned to the currently active layer. A unique LayerName is assigned when a new layer is created.
Data Type:	String
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	LayerIndex Property, LayerPriority Property
Comments:	<p>The name assigned to any layer may be changed using the RenameLayer method.</p> <p>Setting this property to any valid setting will populate the following properties defining this layer:</p> <ul style="list-style-type: none">LayerIndexLayerPriorityLayerVisible <p>All newly created annotation objects will be placed on the layer selected in the LayerName property.</p>

Note: A layer may be selected as the active layer by setting either the **LayerIndex** or the **LayerName** property. It is recommended that any application that tracks or maintains records of layer contents use the **LayerName** property rather than the **LayerIndex** property to identify the layer. The **LayerIndex** of any given layer is subject to change when other layers are created or deleted.

The current layer is not necessarily the top layer for display purposes. The **LayerPriority** property specifies the display order of multiple layers.

LayerPriority Property

Definition: Specifies the display order of the layer on the display window. Layers with a lesser **LayerPriority** value will be displayed above layers with a greater **LayerPriority** value.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read / Write

See Also: LayerVisible Property, LayerIndex Property

Comments: The value of this property applies to the layer selected in the **LayerIndex** or **LayerName** property. The **LayerPriority** is not a unique value for each layer. If multiple layers share a **LayerPriority**, the order in which the layers are displayed is not predictable.

LayerVisible Property

Definition: If False, all annotation objects on this layer will be invisible. If True, the annotation objects on this layer may be visible, depending upon the value of the **ShowNotes** property and the **NoteVisible** property for each object.

Data Type: Boolean

Design Access: Not Available

Runtime Access: Read / Write

See Also: ShowNotes Property, NoteVisible Property, LayerIndex Property

Comments: Any of the current layers may be selected by setting the **LayerIndex** property to a valid value. The active layer may be temporarily hidden by setting the **LayerVisible** property to False. If the annotations are saved, the value of this property is maintained, causing the layer to be invisible when the annotations are loaded again.

Any change in the value of this property will not be reflected in the display window that is linked to this Annotation control until the display window is next drawn. The display window's **Refresh** method may be executed to explicitly freshen the window.

LeftButtonOption Property

Definition: Specifies what action will be performed when the left mouse button is depressed while the cursor is over the Annotation control's display window.

Data Type: Integer (Enumerated)

Design Access: Read / Write

Runtime Access: Read / Write

See Also: RightButtonOption Property, Definitions of the Annotation Types" on page 7

Comments: The valid options for this property are as follows:

- 1 Off
- 2 Pointer
- 3 Highlight
- 4 Redaction
- 5 Text
- 6 Bitmap
- 7 Sticky Note
- 8 Freehand
- 9 Line
- 10 Arrow

1--Off deactivates the Annotation control and prevents the user from creating or modifying annotation objects through the mouse interface.

2--Pointer allows the user to click on any annotation object to select it. The object can then be moved or sized using the mouse.

3--Highlight through *10--Arrow* will create a new annotation object at the mouse cursor when the mouse button is depressed. Refer to "Editing Annotations -- Graphical" on page 25 for details.

Link Property

Definition:	Reports the Link ID calculated for this control at its creation.
Data Type:	String
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	DisplaySource Property, Displaying Annotations on an Image on page 2
Comments:	<p>Each ImageBASIC control is assigned a unique Link ID at its creation. This Link ID can be specified in the ImageDataSource, DisplaySource, RegionSource, and AnnoteSource properties of various ImageBASIC controls. These source properties specify the ImageBASIC control that is supplying information or services to a control.</p> <p>For example, when an Annotation control specifies the Display control that will provide the service of displaying annotation objects, the DisplaySource property is set to the Link ID of the Display control, as shown below:</p> <pre>Annotel.DisplaySource = TMSDispl.Link</pre> <p>As of this writing, an Annotation control cannot be specified in a source property of any other ImageBASIC control.</p>

MouseDown Event

Definition:	Occurs when either mouse button is depressed over an annotation object and the mouse button property is set to anything other than <i>1--Off</i> .																
Parameters:	<table><tr><td>Button</td><td>Reports which mouse button was depressed; 1 for left, 2 for center, 3 for right</td></tr><tr><td>Shift</td><td>Reports if Shift or Control keys were depressed; 1 for Shift, 2 for Control, 3 for both</td></tr><tr><td>X</td><td>Horizontal image pixel coordinate of cursor</td></tr><tr><td>Y</td><td>Vertical image pixel coordinate of cursor</td></tr><tr><td>LayerIndex</td><td>Index of layer on which mouse down occurred</td></tr><tr><td>LayerName</td><td>Name of layer on which mouse down occurred</td></tr><tr><td>NoteIndex</td><td>Index of object at cursor</td></tr><tr><td>NoteName</td><td>Name of object at cursor</td></tr></table>	Button	Reports which mouse button was depressed; 1 for left, 2 for center, 3 for right	Shift	Reports if Shift or Control keys were depressed; 1 for Shift, 2 for Control, 3 for both	X	Horizontal image pixel coordinate of cursor	Y	Vertical image pixel coordinate of cursor	LayerIndex	Index of layer on which mouse down occurred	LayerName	Name of layer on which mouse down occurred	NoteIndex	Index of object at cursor	NoteName	Name of object at cursor
Button	Reports which mouse button was depressed; 1 for left, 2 for center, 3 for right																
Shift	Reports if Shift or Control keys were depressed; 1 for Shift, 2 for Control, 3 for both																
X	Horizontal image pixel coordinate of cursor																
Y	Vertical image pixel coordinate of cursor																
LayerIndex	Index of layer on which mouse down occurred																
LayerName	Name of layer on which mouse down occurred																
NoteIndex	Index of object at cursor																
NoteName	Name of object at cursor																
See Also:	MouseUp Event, Click Event, LeftButtonOption Property																
Comments:	This event will occur each time that a mouse button is depressed. If the mouse button is set to an option that creates a new annotation object, the NoteCreated event will occur before the MouseDown event.																

MouseMove Event

Definition:	Occurs when the mouse cursor moves over any visible annotation object and either the LeftButtonOption or RightButtonOption property is set to any value other than <i>0--Off</i> .	
Parameters:	Button	Reports which mouse button was depressed; 1 for left, 2 for center, 3 for right
	Shift	Reports if Shift or Control keys were depressed; 1 for Shift, 2 for Control, 3 for both
	X	Horizontal image pixel coordinate of cursor
	Y	Vertical image pixel coordinate of cursor
	LayerIndex	Index of layer at cursor
	LayerName	Name of layer at cursor
	NoteIndex	Index of object at cursor
	NoteName	Name of object at cursor
Comments:	Because this event will occur very frequently, any code placed in this event must be kept to a minimum of performance will be significantly degraded.	

MouseUp Event

Definition:	Occurs when either mouse button is released over an annotation object and either the LeftButtonOption or RightButtonOption property is set to any value other than <i>0--Off</i> .	
Parameters:	Button	Reports which mouse button was depressed; 1 for left, 2 for center, 3 for right
	Shift	Reports if Shift or Control keys were depressed; 1 for Shift, 2 for Control, 3 for both
	X	Horizontal image pixel coordinate of cursor
	Y	Vertical image pixel coordinate of cursor
	LayerIndex	Index of layer on which mouse up occurred
	LayerName	Name of layer on which mouse up occurred
	NoteIndex	Index of object at cursor
	NoteName	Name of object at cursor
See Also:	MouseDown Event, Click Event	
Comments:	When the Click event occurs in conjunction with the MouseUp event, the Click event occurs first.	

NoteArrowAngle Property

Definition:	Specifies the degrees of angle between the main line of an Arrow object and each side of the arrowhead.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteArrowLength Property, NoteArrowStyle Property, NoteClass Property, 'Arrow Annotation Class' on page 8
Comments:	This property is valid only when an Arrow object is specified in the NoteIndex or NoteName property. May also be set to a default value for all annotation objects created in the future by setting NoteIndex to 0 (zero) and DefaultClass to 7-- <i>Arrow</i> .

NoteArrowLength Property

Definition:	Specifies the length in pixels of each arm of the arrowhead on an Arrow annotation object.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteArrowAngle Property, NoteArrowStyle Property, NoteClass Property, 'Arrow Annotation Class' on page 8
Comments:	This property is valid only when an Arrow object is specified in the NoteIndex and NoteName property. May also be set to a default value for all annotation objects created in the future by setting NoteIndex to 0 (zero).

NoteArrowLocation Property

Definition:	Specifies the position of the arrow head on an Arrow object.
Data Type:	Enumerated
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteArrowLength Property, NoteArrowStyle Property, NoteClass Property, 'Arrow Annotation Class' on page 8
Comments:	<p>This property is valid only when an Arrow object is specified in the NoteIndex property or when setting defaults for the Arrow class. Valid options for this property are as follows:</p> <ul style="list-style-type: none">0 Head1 Tail2 Both

NoteArrowStyle Property

Definition:	Specifies whether or not the arrowhead on an Arrow annotation object will be filled or hollow.
Data Type:	Integer (Enumerated)
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteArrowAngle Property, NoteArrowLength Property, NoteClass Property, 'Arrow Annotation Class' on page 8
Comments:	This property is valid only when an Arrow annotation object is selected. Valid options for this property are as follows: 0 Filled 1 Hollow

NoteBackColor Property

Definition:	Specifies the color of the background around the text in a Text annotation object.
Data Type:	Long
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteColor Property
Comments:	Applies only to Text annotation objects. This property accepts the RGB color definition such as is returned by Visual Basic's RGB function. The valid range for color is 0 (zero) to FFFFFFFF (hex).

NoteBitmapHandle Property

Definition:	Specifies the handle to the DIB displayed as the annotation object specified in the NoteIndex or NoteName property. Note: This property has been superseded by the NotePicture property. NoteBitmapHandle has been left in the control for backward compatibility.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Write-only
See Also:	NotePicture Property, NoteIndex Property, NoteName Property
Comments:	This property applies only to Bitmap and Sticky Note class objects.

NoteBitmapImageHeight Property

Definition:	Specifies the height of the bitmap annotation in image pixels when it is displayed or printed. Applies only to Bitmap and Sticky Note class objects.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	NoteBitmapImageWidthProperty, NoteBitmapResolutionProperty, "Bitmap Annotation Class" on page 10
Comments:	<p>When a bitmap is displayed as an annotation object, the bitmap can be scaled for the display. The NoteBitmapImageHeight and NoteBitmapImageWidth properties may be used to specify the size of the bitmap.</p> <p>Alternatively, the size of the bitmap may be specified by assigning a resolution to the bitmap through the NoteBitmapResolution property. This resolution will then be applied to scale the bitmap to the underlying image for display.</p>

NoteBitmapImageWidth Property

Definition:	Specifies the width of the bitmap annotation in image pixels when it is displayed or printed. Applies only to Bitmap and Sticky Note class objects.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	NoteBitmapImageHeightProperty, NoteBitmapResolutionProperty, "Bitmap Annotation Class" on page 10
Comments:	<p>When a bitmap is displayed as an annotation object, the bitmap can be scaled for the display. The NoteBitmapImageHeight and NoteBitmapImageWidth properties may be used to specify the size of the bitmap.</p> <p>Alternatively, the size of the bitmap may be specified by assigning a resolution to the bitmap through the NoteBitmapResolution property. This resolution will then be applied to scale the bitmap to the underlying image for display.</p>

NoteBitmapResolution Property

Definition:	Specifies the resolution that the display engine should apply to the annotation bitmap when scaling the bitmap for display. Applies only to Bitmap and Sticky Note class objects.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	NoteBitmapImageHeightProperty, "Bitmap Annotation Class" on page 10
Comments:	<p>When a bitmap is displayed as an annotation object, the size of the bitmap may be specified by assigning a resolution to the bitmap through the NoteBitmapResolution property. This resolution will then be applied to scale the bitmap to the underlying image for display.</p> <p>Alternatively, the bitmap can be scaled for the display using the NoteBitmapImageHeight and NoteBitmapImageWidth properties.</p>

NoteBottom Property

Definition:	Specifies the image pixel position of the rectangle that bounds the annotation object specified in the NoteIndex or NoteName property.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteIndex Property, NoteLeft Property, NoteTop Property
Comments:	<p>This property applies to all annotation types. An annotation object can be moved by changing the value of this property. The value of this property is updated when the NoteIndex or NoteName property is changed to a valid value, when the current annotation object is deleted or moved, and any time that the current object is changed.</p>

NoteClass Property

Definition:	Reports the type of annotation object that is specified in the NoteIndex property.
Data Type:	Integer (Enumerated)
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	NoteIndex Property, Definitions of the Annotation Types on page 7
Comments:	<p>The values that may be reported in this property are as follows:</p> <ul style="list-style-type: none">-1 Layer0 Default (Not Used)1 Reserved (Not Used)2 Unknown3 Highlight4 Redaction5 Text6 Bitmap7 Sticky Note8 Freehand9 Line10 Arrow

NoteColor Property

Definition:	Specifies the color of the annotation object selected in the NoteIndex property.
Data Type:	Long
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteIndex Property
Comments:	<p>The valid range for a normal RGB color is 0 to 16,777,215 (FFFFFF hex). The high byte of a number in this range equals 0; the lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (FF hex).</p> <p>The default color for annotation objects that will be created may be selected by setting NoteIndex to 0 (zero).</p>

NoteCount Property

Definition:	Reports the total number of annotation objects currently loaded in memory.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	NoteIndex Property
Comments:	Any single annotation object may be selected by setting the NoteIndex property to any integer value between 1 (one) and NoteCount , inclusive.

NoteCreated Event

Definition:	Occurs each time that an annotation object is created or loaded from file.										
Parameters:	<table><tr><td>LayerIndex</td><td>Index of layer holding object</td></tr><tr><td>LayerName</td><td>Name of layer holding object</td></tr><tr><td>NoteIndex</td><td>Index of object</td></tr><tr><td>NoteName</td><td>Name of object</td></tr><tr><td>NoteClass</td><td>Reports the type of object</td></tr></table>	LayerIndex	Index of layer holding object	LayerName	Name of layer holding object	NoteIndex	Index of object	NoteName	Name of object	NoteClass	Reports the type of object
LayerIndex	Index of layer holding object										
LayerName	Name of layer holding object										
NoteIndex	Index of object										
NoteName	Name of object										
NoteClass	Reports the type of object										
Comments:	<p>The <i>NoteClass</i> parameter to this event reports an integer value of the type of annotation object that was created, according to the following list:</p> <ul style="list-style-type: none">-1 Layer0 Default (Not Used)1 Reserved (Not Used)2 Unknown3 Highlight4 Redaction5 Text6 Bitmap7 Sticky Note8 Freehand9 Line10 Arrow <p>2--<i>Unknown</i> can occur when loading annotations created by other vendor's software. For complete descriptions of each of the annotation classes, refer to "Definitions of the Annotation Types" on page 7.</p>										

NoteData Property

Definition:	Maintains a user-defined string that is associated with the currently selected annotation object. This string is similar to Visual Basic's Tag property in that the data is maintained but is not used unless specifically coded for by the developer.
Data Type:	String
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	NoteIndex Property, NoteName Property
Comments:	<p>Any single annotation object may be selected by setting the NoteIndex property to any integer value between 1 (one) and NoteCount, inclusive, or by setting the NoteName property to any valid object name, assigned at its creation or set by the application.</p> <p>The Wang annotation format does not allow for the persistence of this property's value, so any information written to this property will be lost if saved in that format. The annotation specification being developed by Diamond Head Software will include the addition of this feature. Until the Annotation control is revised to include the new file specification, the NoteData property will not be saved and the information contained in it may be lost.</p>

NoteDrawMode Property

Definition:	Specifies the scaling of the image files displayed with Bitmap and Sticky Note annotation objects.
Data Type:	Integer (Enumerated)
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteFileName Property, LeftButtonOption Property
Comments:	<p>This property applies only to Bitmap and Sticky Note annotation types. When these object types are displayed, a Windows bitmap file is displayed as the object. The scaling for the display may be set through this property according to these options:</p> <ul style="list-style-type: none">0 Fit Region to Bitmap1 Fit Bitmap to Region2 Clip Bitmap to Region

0--Fit Region to Bitmap causes the Annotation control to calculate the **NoteBottom** and **NoteRight** properties so that the bitmap is displayed at full resolution.

1--Fit Bitmap to Region causes the bitmap to be displayed to fit in the region defined when the object is created. The region may be defined by either selecting a region with the mouse (see the **LeftButtonOption** property for information) or by setting the following properties:

NoteBottom

NoteLeft

NoteRight

NoteTop

2--Clip Bitmap to Region displays the bitmap at full resolution but clips the data to fit the defined region.

NoteFileName Property

- Definition:** Specifies the path and file name of the bitmap file that is displayed as a Bitmap or Sticky Note object.
- Data Type:** String
- Design Access:** Not Available
- Runtime Access:** Read / Write
- See Also:** ShowFileDialog Method, NotePicture Property
- Comments:** The **NoteIndex** or **NoteName** property must specify a Bitmap or Sticky Note object to validate this property. The property may specify either a relative path or an absolute path with the file name. When selecting a bitmap file to display as a Bitmap or Sticky Note object, any valid Windows bitmap may be specified.
- If a bitmap is already available in memory, the **NotePicture** property may be set to the handle to that bitmap to display it.

NoteFillPattern Property

- Definition:** Specifies the pattern used to draw Highlight and Redaction annotation objects.
- Data Type:** Integer (Enumerated)
- Design Access:** Not Available
- Runtime Access:** Read / Write
- See Also:** NoteFillStyle Property, NoteColor Property

Comments: Setting this property to any of the following valid options causes the object selected in **NoteIndex** property to be draw with the appropriate fill pattern:

- 0 Hollow
- 1 Solid
- 2 Horizontal Lines
- 3 Vertical Lines
- 4 Forward Vertical Lines
- 5 Backward Vertical Lines
- 6 Grid
- 7 Diamond

NoteFillStyle Property

Definition: Specifies whether the annotation object specified in the **NoteIndex** property is drawn transparent or opaque.

Data Type: Integer (Enumerated)

Design Access: Not Available

Runtime Access: Read / Write

See Also: NoteFillPattern Property

Comments: This property applies to only Arrow, Freehand and Line objects. The valid options for this property are as follows:

- 0 Opaque
- 1 Transparent

NoteFont Property

Definition: Specifies the attributes of the text in a Text object.

Data Type: See Below

Design Access: Not Available

Runtime Access: Read / Write

See Also: NoteIndex Property, NoteColor Property

Comments: This property specifies the font name, size, bold, italic and strikethrough attributes of the text. These attributes are addressed as subproperties as shown here:

```
Annotel.NoteFont.Name = "Times New Roman"  
Annotel.NoteFont.Size = 12  
Annotel.NoteFont.Bold = True  
Annotel.NoteFont.Italic = False  
Annotel.NoteFont.Strikethrough = False
```

`NoteFont.Name` may be set to the string name of any installed Windows font. If an invalid font is specified, the system default font will be applied.

`NoteFont.Size` may be set to any valid point size for the selected font. If an invalid size is selected, the nearest valid point size will be applied.

`NoteFont.Bold` , `NoteFont.Italic` and `NoteFont.Strikethrough` accept Boolean values.

NoteIndex Property

- Definition:** Specifies the current annotation object, populating the properties that describe the object, and allowing the programmatic modification of the object.
- Data Type:** Integer
- Design Access:** Not Available
- Runtime Access:** Read / Write
- Possible Values:** Any integer from 1 (one) to `NoteCount`
- See Also:** `NoteCount` Property, `NoteName` Property
- Comments:** When **NoteIndex** is set to a valid value, all of the descriptive properties of annotation objects are updated to reflect the selected object. To select any existing annotation object, specify its index which will be between 1 and **NoteCount** inclusive.

Note: If this property is set to 0, the default values for all descriptive properties may be set. Each annotation class -- `Text`, `Highlight`, etc. -- maintains a separate set of defaults. Therefore, when setting default values, set the **DefaultClass** property to specify which type of object is being defined. Not all of the properties may be modified by setting default values. Those capable of being modified are marked in the table below with an asterisk *.

Following are the descriptive properties that specify the characteristics of the annotation object specified in the **NoteIndex** property.

<code>NoteArrowAngle</code> *	<code>NoteLineStyle</code> *
<code>NoteArrowLocation</code> *	<code>NoteName</code>
<code>NoteArrowLength</code> *	<code>NotePointCount</code>
<code>NoteArrowStyle</code> *	<code>NotePointIndex</code>
<code>NoteBackColor</code>	<code>NotePointX</code>
<code>NoteBottom</code>	<code>NotePointY</code>
<code>NoteClass</code>	<code>NotePrint</code>

NoteColor *	NoteRight
NoteDrawMode *	NoteSave
NoteFileName	NoteSelected
NoteFillPattern *	NoteShape *
NoteFillStyle *	NoteText *
NoteFont *	NoteThickness *
NoteLeft	NoteTop
	NoteVisible

NoteLeft Property

Definition:	Specifies the image pixel coordinate of the left edge of a rectangle that bounds the current annotation object.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
Possible Values:	Any integer from 0 (zero) to image width
See Also:	NoteTop Property, NoteRight Property
Comments:	This property applies to all annotation types. Changing the value of this property will move or size the current object, but the change will not be shown until the display window is refreshed.

NoteLineStyle Property

Definition:	Specifies the style of the lines in the current annotation object. Applies to Line, Arrow, and Freehand objects.										
Data Type:	Integer (Enumerated)										
Design Access:	Not Available										
Runtime Access:	Read / Write										
Possible Values:	<table> <tr> <td>0</td><td>Solid</td></tr> <tr> <td>1</td><td>Dash</td></tr> <tr> <td>2</td><td>Dot</td></tr> <tr> <td>3</td><td>Dash Dot</td></tr> <tr> <td>4</td><td>Dash Dot Dot</td></tr> </table>	0	Solid	1	Dash	2	Dot	3	Dash Dot	4	Dash Dot Dot
0	Solid										
1	Dash										
2	Dot										
3	Dash Dot										
4	Dash Dot Dot										
See Also:	NoteThickness Property										
Comments:	<p>The line size is specified through the NoteThickness property. This integer property specifies the number of pixels across the width of the line that is drawn.</p> <p>Only when the line is a single point wide, i.e., when the NoteThickness property equals 1 (one), the style of line that is</p>										

drawn (solid, dotted, etc.) is specified through the **NoteLineStyle** property.

NoteName Property

Definition: A unique string assigned to each annotation object at its creation. When set to the name of any currently loaded annotation object, the **NoteIndex** property is updated and all descriptive properties are populated to reflect the selected object.

Data Type: String

Design Access: Not Available

Runtime Access: Read / Write

Possible Values: Any valid string up to 16 characters long

See Also: NoteIndex Property, RenameNote Method

Comments: The name assigned to an annotation object may be changed using the **RenameNote** method. When this property is set to any valid value, the following descriptive properties are updated:

NoteArrowAngle	NoteLineStyle
NoteArrowLocation	NoteName
NoteArrowLength	NotePointCount
NoteArrowStyle	NotePointIndex
NoteBackColor	NotePointX
NoteBottom	NotePointY
NoteClass	NotePrint
NoteColor	NoteRight
NoteDrawMode	NoteSave
NoteFileName	NoteSelected
NoteFillPattern	NoteShape
NoteFillStyle	NoteText
NoteFont	NoteThickness
NoteLeft	NoteTop
	NoteVisible

NotePicture Property

Definition:	Specifies the current Windows handle to the DIB being displayed as the current annotation object. Applies only to Bitmap and Sticky Note class objects.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	NoteFileName Property, NoteIndex Property, NoteName Property, NoteFileName Property
Comments:	<p>This property is equivalent to the Picture property of the Picture Box and Image controls available with Visual Basic. The Annotation control supports only bitmaps, so at run time, you can set this property using the Visual Basic LoadPicture function on a bitmap.</p> <p>The NoteFileName property may also be used to specify a bitmap for display as one of these object types.</p>

NotePointCount Property

Definition:	Specifies the total number of points that describe a Freehand, Line or Arrow annotation object.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read-only
Possible Values:	Any positive integer
See Also:	NotePointIndex Property
Comments:	When the object is displayed, the control connects the points as smoothly as possible.

NotePointIndex Property

Definition:	Specifies a single point of the series of points that describes a Freehand, Line or Arrow annotation object.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
Possible Values:	Any integer from 1 (one) to NotePointCount
See Also:	NotePointCount Property, NoteThickness Property, NotePointX Property, NotePointY Property, NoteClass Property
Comments:	When this property is set to a valid value, the NotePointX and NotePointY properties are populated with the values that describe the selected point. The NoteThickness property is also

updated, but it must be maintained at the same value through all points in a single object.

NotePointX Property

Definition: Specifies the X value of the coordinate of the point specified in the **NotePointIndex** property, in image pixels. Applies only to Freehand, Line and Arrow annotation objects.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read-only

Possible Values: Any integer from 0 (zero) to image width

See Also: NotePointY Property, NotePointIndex Property

Comments: The **NotePointX** and **NotePointY** properties report the coordinate of a single point in an annotation object. The coordinate of each point in the object is stated in image pixels.

NotePointY Property

Definition: Specifies the Y value of the coordinate of the point specified in the **NotePointIndex** property, in image pixels. Applies only to Freehand, Line and Arrow annotation objects.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read-only

Possible Values: Any integer from 0 (zero) to image height

See Also: NotePointX Property, NotePointIndex Property

Comments: The **NotePointX** and **NotePointY** properties report the coordinate of a single point in an annotation object. The coordinate of each point in the object is stated in image pixels.

NotePrint Property

Definition: If True, the annotation object will be printed if loaded when the underlying image is printed through the display window's regular print methods. Applies to all annotation types.

Data Type: Boolean

Design Access: Not Available

Runtime Access: Read / Write

Possible Values: True (Default)
False

See Also: NoteSave Property

Comments: When the display window that is specified in the Annotation control's **DisplaySource** property is printing an image, each annotation object that is displayed will be printed on the image if the object's **NotePrint** property is True.

NotePriority Property

Definition: Specifies the order in which stacked annotation objects are displayed.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read / Write

Possible Values: Any positive integer
Default: 0

See Also: NoteIndex Property, LayerPriority Property

Comments: When multiple annotation objects overlap, the object with a lower **NoteIndex** value will be displayed on top of other objects. The effect of this is that older objects are drawn on top of newer objects because as each new object is created, it is assigned the next highest **NoteIndex**.

The **NotePriority** property may be modified for the stacked objects to change their display order. As each object is created, it is assigned a default **NotePriority** of 0 (zero), the highest priority.

Higher **NotePriority** values cause the object to be drawn farther back in the stack of objects. Changing an object's **NotePriority** can also change the **NoteIndex** assigned to that object, so all tracking of individual objects should be based on the **NoteName**.

NoteRight Property

Definition: Specifies the image pixel coordinate of the right edge of a rectangle that bounds the current annotation object.

Data Type: Integer

Design Access: Not Available

Runtime Access: Read / Write

Possible Values: Any integer from 0 (zero) to image width

See Also: NoteLeft Property, NoteTop Property

Comments: This property applies to all annotation types. Changing the value of this property will move or size the current object.

NoteRotation Property

Definition:	Specifies the orientation of Text and Bitmap class objects.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
Possible Values:	0 (Default) 90 180 270
See Also:	NoteIndex Property
Comments:	As Text and Bitmap objects are created, they will be drawn on the display upright. If the image is rotated, the objects will stay upright. After selecting an object by setting the NoteIndex or NoteName property to a valid value, the NoteRotation property may be set for these object types to display and print then in the proper orientation.

NoteSave Property

Definition:	If True when annotations are saved through the AnnoteSource property or through the WriteNotes method, the object will be written to file. If False, the object will not be saved.
Data Type:	Boolean
Design Access:	Not Available
Runtime Access:	Read / Write
Possible Values:	True (Default) False
See Also:	WriteNotes Method, AnnoteSource Property
Comments:	Applies to all annotation types. Each individual annotation object will be saved only if this property is True, the default. The current value of this property reflects the object specified in the NoteIndex or NoteName property.

NoteSelected Property

Definition:	If True, the object will be drawn as selected i.e., outlined with grab handles. If False, the object will be drawn normally.
Data Type:	Boolean
Design Access:	Not Available
Runtime Access:	Read / Write
Possible Values:	True False (Default)
See Also:	NoteIndex Property, NoteName Property
Comments:	Applies to all annotation types. The current value of this property reflects the object specified in the NoteIndex or NoteName property.

NoteShape Property

Definition:	Specifies the shape of a Highlight or Redaction object.
Data Type:	Integer (Enumerated)
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteFillPattern Property, NoteIndex Property
Comments:	When a Highlight or Redaction object is displayed, its shape may be specified by setting this property to one of these options: 0 Rectangle 1 Rounded Rectangle 2 Ellipse

NoteText Property

Definition:	Specifies the text string that is in a Text or Sticky Note annotation object.
Data Type:	String
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteFont Property, NoteIndex Property
Comments:	The text string that is displayed as a Text annotation object and the text that is contained in a Sticky Note annotation object are reported and set through this property.

NoteThickness Property

Definition:	Specifies the line thickness in image pixels for Arrow, Line and Freehand objects.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteLineStyle Property
Comments:	This value may be set to any non-zero integer value to specify the line thickness of the current annotation object. The NoteLineStyle property is valid only when NoteThickness is 1.

NoteTop Property

Definition:	Specifies the image pixel coordinate of the top edge of a rectangle that bounds the current annotation object.
Data Type:	Integer
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NoteBottom Property, NoteLeft Property, NoteIndex Property, NoteName Property
Comments:	This property applies to all annotation types. Changing the value of this property will move or size the current object.

NoteVisible Property

Definition:	If True, the current annotation object, as specified in the NoteIndex property, will be visible, as long as the LayerVisible property of the object's layer and the ShowNotes property are also True.
Data Type:	Boolean
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	LayerVisible Property, ShowNotes Property
Comments:	<p>When one or more annotation objects are loaded, either from file or newly created, an object will be visible if the following properties are True:</p> <p> ShowNotes, LayerVisible, NoteVisible</p> <p>These properties are shown in priority order. That is, the ShowNotes property must be True for <i>any</i> annotations to be visible, and the LayerVisible property must be True for any annotation objects on that layer to be visible.</p>

PrintNotes Property

Definition:	If True, annotation objects can be printed with the underlying image. Each object will be printed if its own NotePrint property is True. If False, annotation objects will not be printed with the underlying image.
Data Type:	Boolean
Design Access:	Read / Write
Runtime Access:	Read / Write
See Also:	NotePrint Property
Comments:	When the image on which one or more annotation objects are displayed is printed through the display control PrintImage or PrintRegion method, the annotations on that region will be merged into the print output only if the PrintNotes property is True. Each individual object is also described by NotePrint property which may be set to False to disable printing of that particular object.

ReadNotes Method

Definition:	Loads all annotation objects from any supported annotation file format. Does not read annotation data from image file headers.				
Parameters:	<table><tr><td>filename</td><td>Name of file to load</td></tr><tr><td>format</td><td>Enumerated file format specification</td></tr></table>	filename	Name of file to load	format	Enumerated file format specification
filename	Name of file to load				
format	Enumerated file format specification				
Syntax:	<code>Annotel.ReadNotes filename, format</code>				
Return Value:	Number of annotation objects on success -1 on failure				
Data Type:	Long				
See Also:	WriteNotes Method				
Comments:	<p>The <i>filename</i> parameter may include an absolute or relative path. The <i>format</i> parameter to this event accepts the following enumerated options:</p> <ol style="list-style-type: none">0 <i>Autodetect</i>; default value1 <i>DHS</i>; not yet implemented2 <i>Wang</i>; format used by earlier ImageBASIC 3.0 revisions3 <i>Pixel</i>; not implemented; format used by ImageBASIC 2.x4 <i>Watermark</i>; not implemented <p>When annotations are loaded from file, the NoteCreated event occurs once for each object as it is read into memory. The display control to which this Annotation control is linked will not show the newly loaded objects until it is refreshed.</p>				

RenameLayer Method

Definition:	Changes the name assigned to the current layer as selected in the LayerIndex property.
Parameters:	Name New name for the layer
Syntax:	<code>Annotel.RenameLayer <i>newname</i></code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	LayerIndex Property, LayerName Property, RenameNote Method
Comments:	A layer name is limited to 18 characters. Any printable character may be included.

RenameNote Method

Definition:	Changes the name assigned to a particular annotation object. The object being renamed must be specified in the NoteIndex property.
Parameters:	NewItem New name for the annotation object
Syntax:	<code>Annotel.RenameNote <i>newname</i></code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	NoteIndex Property, NoteName Property, RenameLayer Method
Comments:	The name assigned to an annotation object is limited to 18 characters, and any of the printable characters may be included. Note: It is recommended that any application which requires the accurate programmatic specification of particular annotation objects should specify objects by the NoteName rather than the NoteIndex . This recommendation is necessary because the NoteIndex assigned to a particular object is subject to change with the creation and deletion of other annotation objects.

RightButtonOption Property

Definition:	Specifies what action will be performed when the right mouse button is depressed while the cursor is over the Annotation control's display window.
Data Type:	Integer (Enumerated)
Design Access:	Read / Write
Runtime Access:	Read / Write

See Also: LeftButtonOption Property, Definitions of the Annotation Types" on page 7

Comments: The valid options for this property are as follows:

- 1 Off
- 2 Pointer
- 3 Highlight
- 4 Redaction
- 5 Text
- 6 Bitmap
- 7 Sticky Note
- 8 Freehand
- 9 Line
- 10 Arrow

1--Off deactivates the Annotation control and prevents the user from creating or modifying annotation objects through the mouse interface.

2--Pointer allows the user to click on any annotation object to select it. The object can then be moved or sized using the mouse.

3--Highlight through *10--Arrow* will create a new annotation object at the mouse cursor. Refer to Editing Annotations -- Graphical" on page 25 for details on this process.

SendToBack Method

Definition: Changes the display priority of the current annotation object so that it is displayed below all other objects on that layer.

Parameters: None

Syntax: `Annotel.SendToBack`

Return Value: 0 on success
-1 on failure

Data Type: Long

See Also: BringToFrontMethod, NotePriority Property

Comments: Executing this method will alter the current object's **NotePriority** to equal **NoteCount**+ 1 and **NoteIndex** to equal **NoteCount** so that the object is at the bottom of the display stack. This can cause other objects **NotePriority** and **NoteIndex** values to change as well.

ShowFileDialog Method

Definition:	Displays an input message dialog requesting the file name of a bitmap image to use in a Bitmap or Sticky Note annotation object.
Parameters:	None
Syntax:	<code>Annotel.ShowFileDialog</code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	NoteClass Property, NoteIndex Property
Comments:	<p>The NoteIndex or NoteName property must be set to specify a Bitmap or Sticky Note object in order to successfully execute this method. When the method is executed, a dialog is displayed requesting the file name of a Windows BMP file to insert in the specified object. If the selected object has already been assigned a file, the current selection will be shown in the dialog.</p> <p>A typical application of this method is in the NoteCreated event if the application allows the user to graphically define new objects. In this event, when a Bitmap or Sticky Note is created, as reported in the NoteClass parameter to the event, this method is executed and the operator is prompted to select a file for display.</p>

ShowNotes Property

Definition:	If True, annotation objects currently in memory will be displayed if their respective LayerVisible and NoteVisible properties are True.
Data Type:	Integer (Boolean)
Design Access:	Not Available
Runtime Access:	Read / Write
See Also:	NotePrint Property, LayerVisible Property, NoteVisible Property
Comments:	<p>Setting ShowNotes to False will remove all annotations from the display, but will not otherwise modify the objects. Setting ShowNotes back to True will again display the annotations.</p> <p>When ShowNotes is True, each layer may be individually shown or hidden through the LayerVisible property. When both ShowNotes and LayerVisible are True, individual objects may be shown or hidden through the NoteVisible property.</p>

ShowTextDialog Method

Definition:	Displays an input dialog requesting the text string to use in a Text annotation object.
Parameters:	None
Syntax:	<code>Annotel.ShowTextDialog</code>
Return Value:	0 on success -1 on failure
Data Type:	Long
See Also:	NoteClass Property, NoteText Property, NoteIndex Property, NoteCreated Event
Comments:	<p>A Text or Sticky Note object must be specified in the NoteIndex or NoteName property in order to successfully execute this method. The string that currently defines the object will be displayed in a dialog box, and the operator is offered the options of changing the text or exiting without saving the changes. The dialog displays the name of the current object as the title in the window.</p> <p>A typical application of this method is in the NoteCreated event if the application allows the user to graphically define new objects. In this event, when a Text or Sticky Note is created, as reported in the <i>NoteClass</i> parameter to the event, this method is executed and the operator is prompted to type in the text for display.</p>

WriteNotes Method

Definition:	Writes an annotation file containing the currently loaded annotation objects.				
Parameters:	<table><tr><td>filename</td><td>File name of the annotation file</td></tr><tr><td>format</td><td>Enumerated specification of the type</td></tr></table>	filename	File name of the annotation file	format	Enumerated specification of the type
filename	File name of the annotation file				
format	Enumerated specification of the type				
Syntax:	<code>Annotel.WriteNotes filename, format</code>				
Return Value:	0 on success -1 on failure				
Data Type:	Long				
See Also:	ReadNotes Method				
Comments:	<p>The <i>filename</i> parameter may include an absolute or relative path. The <i>format</i> parameter to this event accepts the following enumerated options:</p> <ol style="list-style-type: none">1 <i>DHS</i>; Not yet implemented2 <i>Wang</i>; Default value; format used by earlier ImageBASIC 3.x revisions				

When annotation objects are written to a separate annotation file, all of the position and descriptive information of the objects is saved. Any annotation file written by this method may be into the Annotation control using the **ReadNotes** method.

Note: The files created by this method are not inherently associated with any image file and may be loaded and displayed with any valid image. The association of image files and annotation files is left to the developer's discretion.

Index

A

- AboutBox Method 49
- Active Property 49
- Annotation Classes 7
 - Arrow 8
 - Bitmap 10
 - Freehand 12
 - Highlight 14
 - Line 15
 - Pointer 19
 - Redaction 17
 - Sticky Note 20
 - Text 21
- Annotation Modes
 - Freehand 12
- AppendPoint Method 33, 50
- Arrow Annotation Class 8
- Arrow Class Properties
 - NoteArrowAngle 9, 69
 - NoteArrowLength 9, 69
 - NoteArrowLocation 8, 69
 - NoteArrowStyle 9, 70

B

- Bitmap
 - NoteBitmapHandle Property 70
 - NoteBitmapImageHeight Property 71
 - NoteBitmapImageWidth Property 71
 - NoteBitmapResolution Property 72
- Bitmap Annotation Class 10
- Bitmap File
 - NoteFileName Property 76
- Bitmap for Display
 - ShowFileDialog Method 91
- BringToFront Method 51

C

- Class (Type)
 - NoteClass Property 73
- Classes of Annotations 7
- ClearNotes Method 51
- Click Event 51
- Color
 - NoteColor Property 73

- Color of Annotations
 - NoteBackColor Property 70
 - NoteColor Property 9, 13, 14, 16, 17, 21
- Combining Annotation Files, MergeNotes Method 67

Count

- NoteCount Property 74
- NoteData Property 75
- CreateArrow Method 32, 52
- CreateBitmap Method 32, 52
- CreateFreehand Method 33, 53
- CreateHighlight Method 33, 54
- CreateLayer Method 34, 54
- CreateLine Method 55
- CreateRedaction Method 56
- CreateStickyNote Method 34, 58
- CreateText Method 35, 59
- Creating Annotations
 - LeftButtonOption Property 8, 10, 12, 20, 23, 25, 66
 - RightButtonOption Property 8, 10, 12, 14, 20, 23, 25, 89

D

- DbClick Event 60
- DefaultClass Property 24, 60
- Defaults, Setting 24
 - DefaultClass Property 60
- Defaults, Standard 30
- DeleteLayer Method 61
- DeleteNote Method 61
- DeletePoint Method 62
- Deleting Annotations
 - ClearNotes Method 51
 - DeleteLayer Method 61
 - DeleteNote Method 61
- DIB Handle
 - NotePicture Property 82
- Display 51, 90
- Display Order, LayerPriority Property 65
- DisplaySource Property 2

E

- Editing Modes
 - Freehand 12
- Enabling Printing of Notes
 - NotePrint Property 41
 - PrintNotes Property 41, 88

Error Event 63

Events

Click 51

DbClick 60

Error 63

MouseDown 67

MouseMove 68

MouseUp 68

F

Fill Pattern

NoteFillPattern Property 76

Fonts

NoteFont Property 77

Freehand Annotation Class 12

Freehand Annotations

NotePointCount Property 82

NotePointIndex Property 82

NotePointX Property 83

NotePointY Property 83

Freehand Class, Deleting Points

DeletePoint Method 62

Freehand Mode 12

H

Handle to DIB

NotePicture Property 82

Hiding Annotations 20

Highlight Annotation Class 14

I

ImageDataSource Property 62

Introduction 1

L

LayerCount Property 63

LayerIndex Property 64

LayerName Property 64

LayerPriority Property 65

LayerVisible Property 65

LeftButtonOption Property 8, 10, 12, 20, 23,
25, 66

Licensing

Active Property 49

Line Annotation Class 15

Line Size

NoteThickness Property 87

Line Style

NoteLineStyle Property 79

Linking

DisplaySource Property 2

Linking Controls

ImageDataSource Property 62

Load Time Improvement

Active Property 49

Loading Saved Annotations

ReadNotes Method 88

M

MergeNotes Method 67

Methods

AboutBox 49

AppendPoint 33, 50

BringToFront 51

ClearNotes 51

CreateArrow 32, 52

CreateBitmap 32, 52

CreateFreehand 33, 53

CreateHighlight 33, 54

CreateLayer 34, 54

CreateLine 55

CreateRedaction 56

CreateStickyNote 58

CreateStickyNote 34

CreateText 35, 59

DeleteLayer 61

DeleteNote 61

DeletePoint 62

MergeNotes 67

ReadNotes 88

RenameLayer 89

RenameNote 89

SendToBack 90

ShowFileDialog 91

ShowTextDialog 92

WriteNotes 92

MouseDown Event 67

MouseMove Event 68

MouseUp Event 68

N

Name, Changing

RenameLayer Method 89

RenameNote Method 89

NoteArrowAngle Property 9, 69

- NoteArrowLength Property 9, 69
- NoteArrowLocation Property 8, 69
- NoteArrowStyle Property 9, 70
- NoteBackColor Property 70
- NoteBitmapHandle Property 70
- NoteBitmapImageHeight Property 71
- NoteBitmapImageWidth Property 71
- NoteBitmapResolution Property 72
- NoteBottom Property 72
- NoteClass Property 73
- NoteColor Property 9, 13, 14, 16, 17, 21, 73
- NoteCount Property 74
- NoteData Property 75
- NoteDrawMode Property 11, 75
- NoteFileName Property 76
- NoteFillPattern Property 76
- NoteFillStyle Property 77
- NoteFont Property 77
- NoteIndex Property 24, 78
- NoteLeft Property 79
- NoteLineStyle Property 79
- NoteName Property 81
- NotePicture Property 82
- NotePointCount Property 82
- NotePointIndex Property 82
- NotePointX Property 83
- NotePointY Property 83
- NotePrint Property 41, 83
- NotePriority Property 84
- NoteRight Property 84
- NoteRotation Property 85
- NoteSave Property 85
- NoteSelected Property 86
- NoteShape Property 86
- NoteText Property 86
- NoteThickness Property 8, 12, 15, 79, 87
- NoteTop Property 87
- NoteVisible Property 87

P

- Pointer Annotation Class 19
- PopUp Annotations
 - NoteText Property 86
- Position
 - NoteBottom Property 72
 - NoteLeft Property 79
 - NotePriority Property 84
 - NoteRight Property 84
 - NoteRotation Property 85

- NoteTop Property 87
- Printing Annotations
 - NotePrint Property 83
- PrintNotes Property 41, 88
- Programming New Objects
 - AppendPoint Method 33, 50
 - CreateArrow Method 32, 52
 - CreateBitmap Method 32, 52
 - CreateFreehand Method 33, 53
 - CreateHighlight Method 33, 54
 - CreateLayer Method 34, 54
 - CreateLine Method 55
 - CreateRedaction Method 56
 - CreateStickyNote Method 34, 58
 - CreateText Method 35, 59
- Properties
 - Active 49
 - Default 24
 - DefaultClass 60
 - DisplaySource 2
 - ImageDataSource 62
 - LayerCount 63
 - LayerIndex 64
 - LayerName 64
 - LayerPriority 65
 - LayerVisible 65
 - LeftButtonOption 8, 10, 12, 20, 23, 25, 66
 - NoteArrowAngle 9, 69
 - NoteArrowLength 9, 69
 - NoteArrowLocation 8, 69
 - NoteArrowStyle 9, 70
 - NoteBackColor 70
 - NoteBitmapHandle 70
 - NoteBitmapImageHeight 71
 - NoteBitmapImageWidth 71
 - NoteBitmapResolution 72
 - NoteBottom 72
 - NoteClass 73
 - NoteColor 9, 13, 14, 16, 17, 21, 73
 - NoteCount 74
 - NoteData 75
 - NoteDrawMode 11, 75
 - NoteFileName 76
 - NoteFillPattern 76
 - NoteFillStyle 77
 - NoteFont 77
 - NoteIndex 24, 78
 - NoteLeft 79
 - NoteLineStyle 79

- NoteName 81
- NotePicture 82
- NotePointCount 82
- NotePointIndex 82
- NotePointX 83
- NotePointY 83
- NotePrint 41, 83
- NotePriority 84
- NoteRight 84
- NoteRotation 85
- NoteSave 85
- NoteSelected 86
- NoteShape 86
- NoteText 86
- NoteThickness 8, 12, 15, 79, 87
- NoteTop 87
- NoteVisible 87
- PrintNotes 41, 88
- RightButtonOption 8, 10, 12, 14, 20, 23, 25, 89
- ShowNotes 91

R

- ReadNotes Method 88
- Redaction Annotation Class 17
- RenameLayer Method 89
- RenameNote Method 89
- RightButtonOption Property 8, 10, 12, 14, 20, 23, 25, 89

S

- Saving
 - NoteSave Property 85
- Saving and Loading Annotations
 - Separate Files 45
 - TIFF Files 42
- Saving Files
 - ImageDataSource Property 62
- Saving Separate Annotation Files 45
- Saving to File
 - WriteNotes Method 92
- Scaling Bitmap
 - NoteDrawMode Property 11, 75
- Scaling Bitmaps
 - NoteBitmapImageHeight Property 71
 - NoteBitmapImageWidth Property 71
 - NoteBitmapResolution Property 72
- Selecting a Layer

- LayerIndex Property 64
- LayerName Property 64
- Selecting an Annotation
 - NoteName Property 81
- Selecting an Object
 - NoteIndex Property 24
- Selecting Annotation Modes
 - Dialog Box 38
- Selecting Annotations
 - NoteIndex Property 78
- SendToBack Method 90
- Shape
 - NoteShape Property 86
- ShowFileDialog Method 91
- Showing Annotations
 - LayerVisible Property 65
 - NoteVisible Property 87
 - ShowNotes Property 91
- ShowNotes Property 20, 91
- ShowTextDialog Method 92
- Sticky Note Annotation Class 20

T

- Text Annotation Class 21
- Text Annotations
 - NoteText Property 86
- Text for Display
 - ShowTextDialog Method 92
- Thickness
 - NoteThickness Property 8, 12, 15, 79
- Timing of Licensing Verification
 - Active Property 49
- Transparency
 - NoteFillStyle Property 77
- Types of Annotations 1

V

- Version Information
 - AboutBox Method 49
- Viewing Annotations 20

W

- WriteNotes Method 92