ViewONE Annotations Installation, HTML and JavaScript Manual

Version 3.0

Last Updated: 21st January 2005

Email: info@daeja.com
Web site: http://www.daeja.com

Contents

Introduction

ViewONE is a Java applet that extends your web browser so that you can view, zoom, magnify, scroll, pan, rotate and print your images and image documents quickly and easily.

This document is the Applet Annotations Installation, HTML and JavaScript Manual. This document is designed to be used in conjunction with the ViewONE HTML and Installation Manual and the ViewONE JavaScript API Manual.

For further information about ViewONE please consult the following documents…

- Applet User Manual
- ViewONE HTML and Installation Manual
- ViewONE JavaScript API Manual
- ViewONE Annotations User Manual
- Various White Papers for in-depth analysis of specific areas
- Latest ViewONE release notes found at http://www.daeja.com/pub/start/release.html

ViewONE Annotations Overview

Annotation data is kept separate from image files

Unlike most viewers, ViewONE separates annotation data from the image files. This has several significant advantages when dealing with annotating documents in a web environment.

- By keeping the annotations separate from image files (and in text form) then the download and upload of annotations can also be kept separate.
- This in turn means quicker repeat download and upload of annotations because the amount of data is much less (than if it was attached to the image file), it also means a true web based annotation editing system can be deployed because there are no longer large delays receiving and sending image to a from the server.
- And last (but not least), it also means it is possible for annotations to be held in a database, and therefore searched upon independently of image files.

Server Component

The implementation of annotations requires both a client-side and server-side component.

The client-side component is ViewONE, the server side component is required for saving annotations and should be written/provided by whoever implements the system.

This manual provides the necessary specifications to permit writing of this component.

This document assumes you already know how to install and use the applet (without annotations). If this is the first time you have installed ViewONE, please read the HTML manual prior to embarking on annotations.

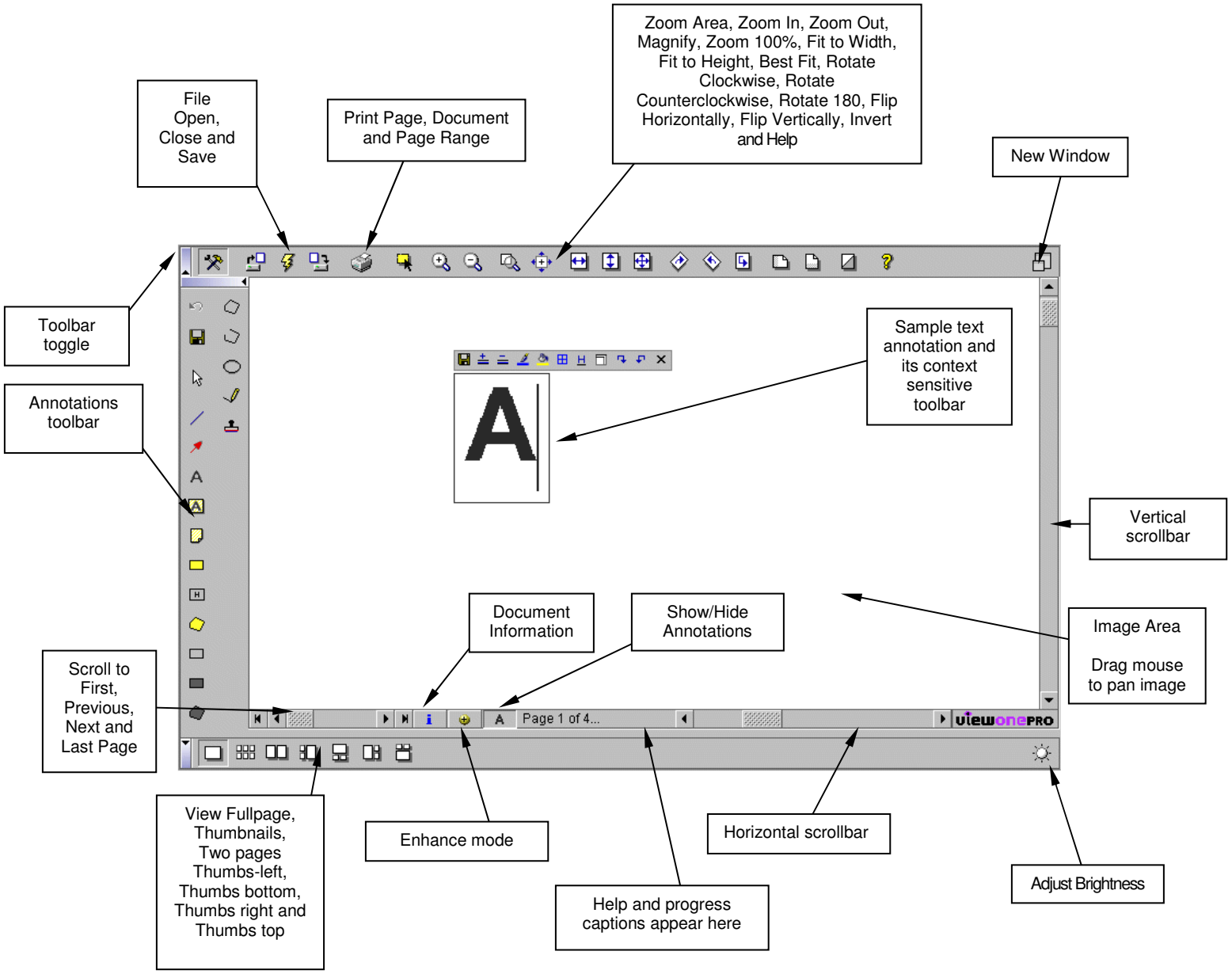
For sample server-side components (written in CGI/Perl and ASP/VBScript) please see Appendix E or samples available at www.daeja.com/pub/start/downloads.html

Annotations Licensing

To use the ViewONE annotations module you will require a license file that has the annotations module enabled.

If you are unsure if your version is licensed for annotations, then when you have installed ViewONE and have it running, hold down the CTRL and SHIFT keys then click the mouse in the applet's image display area. This should bring up a box containing the line "Extensions: Annotations". If this extension is not displayed, then you will need to purchase it (please see www.daeja.com/pub/start/buy.html).

ViewONE and the Annotations User Interface



Applet Parameters

The following list describes parameters that can be included in HTML code using the standard HTML Applet parameter specification:

<PARAM NAME="*name*" value="*value*">

Where *name* is replaced by the parameter name and *value* is replaced by the parameter value.

All parameters are specified as strings.

Filenames and hyperlink addresses are expressed using the URL address format (Uniform Resource Locator), e.g. "http://mysite/myscript.pl". If any part of the address before "myscript.pl" is not included then the applet will assume a base address which is the same as the applet location (the codebase).

URLs specified to the applet's parameters can be relative or absolute. For example, "../docs/image1.ant" (relative), "http://www.daeja.com/docs/image1.ant" (absolute).

With the exception of filenames and hyperlink addresses, all parameters are case insensitive.

In this manual all parameters relate to annotations.

ViewONE Version 3 Security Change

ViewONE version includes a change that prevents unauthorized use of some JavaScript methods. These are any methods that can be used to change the behavior of ViewONE including some methods relating to the Annotations JavaScript API.

These methods are disabled by default, and to enable them you need to use the new HTML paramater as follows…

<PARAM NAME="annotationJavascriptExtensions" value="true">

The methods that are affected are as follows:

- reloadAnnotations()
- setAnnotateEdit(*boolean*)
- int getNumAnnotations(*type*, *page*)
- string getAnnotationLabels(*type*, *page*)
- string getAnnotationLabels(*label*)
- addAnnotation(*annotationProperties*)
- modifyAnnotation(*label*, *annotationProperties*)
- deleteAnnotation(*label*)
- deleteAllAnnotations(*type*, *page*)
- startAnnotation(*type*)

Opening and saving annotations

Parameter: **annotationFile**

<PARAM NAME="annotationFile" value="../docs/p1.ant">
<PARAM NAME="annotationFile" value="http://www.mysite.com/myscript.pl?userID=12&docID=8791">

ViewONE requires an annotations definition file which contains a list of the annotations to display (and their associated parameters). This parameter defines that file.

It specifies the location of the annotations definition file relative to the applet's codebase. The value can also specify a server-side object (CGI, EXE, ASP, etc.) that streams an annotations file to the applet.

Parameters may be added to the URL, such as user ID or document ID, in order to provide user or document specific annotations. If you use a server-side object to supply/stream this file then it is up to this object to handle any parameters you may include.

The annotations file itself must conform to the ViewONE annotations file specification described later in this manual (see page 65).

If you are unsure whether ViewONE has processed your annotations file correctly then you can enable the annotations "debug" output (see page 43).

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Parameter: **annotationSavePost**

<PARAM NAME="annotationSavePost" value="http://www.mysite.com/annotationsave.dll">

Specifies the location of a server-side object (CGI, EXE, ASP, JSP, etc.) that handles the saving of annotations created by the viewer. It is up to whoever implements the system to supply the object, which must conform to the specification for ViewONE annotations save POST objects (see page 93).

If a relative address is used, the location it defines is relative to the applet's codebase.

Parameters may be added to the POST data, such as user ID or document ID to send user or document specific annotations, by using the "annotationPostPrefix" parameter. ViewONE will then tag on its own annotations data.

If you are unsure what ViewONE is sending through to the object then you can enable the annotations "debug" output (see page 43).

Note that the object will only be called if the document displayed has been retrieved through a web server. For example, if a document is retrieved through the file:/// protocol then the object will not be called when the user hits the annotations save button.

Parameter: **annotationPostPrefix**

<PARAM NAME="annotationPostPrefix" value="p1=value1&p2=value2">

This parameter works with the "annotationSavePost" to specify parameters to be added to the POST data for the annotationSavePost feature (see above). The above example demonstrates prefixing "p1=value1" and "p2=value2" to the POST data.

The POST object should then be able to parse the data posted to extract all the parameters, in this case p1 and p2, plus the usual size, numdata, data01, data02, etc. (see page 93).

Parameter: **annotationSaveServlet**

<PARAM NAME="annotationSaveServlet"
value="http://www.mysite.com/servlet/annotationsave.class?userID=12&docID=8791">

Specifies the location of a servlet that handles the saving of annotations created by the viewer. It is up to whoever implements the system to supply the servlet, which must conform to the specification for ViewONE annotations save servlets (see page 94).

If a relative address is used, the location it defines is relative to the applet's codebase.

Parameters may be added to the URL (as in the "annotationsFile" parameter), such as user ID or document ID, in order to send user or document specific annotations.

If you are unsure what ViewONE is sending through to the servlet then you can enable the annotations "debug" output (see page 43).

Note that the servlet will only be called if the document displayed has been retrieved through a web server. For example, if a document is retrieved through the file:/// protocol then the servlet will not be called when the user hits the annotations save button.

Parameter: **annotationSave**

<PARAM NAME="annotationSave" value="http://www.mysite.com/myscript.pl?userID=12&docID=8791&">

Specifies the location of a server-side object (CGI, EXE, ASP, JSP, etc.) that handles the saving of annotations created by the viewer using multiple HTTP:GET calls. It is up to whoever implements the system to supply the object, which must conform to the specification for ViewONE annotations save GET objects (see page 95).

If a relative address is used, the location it defines is relative to the applet's codebase.

Parameters may be added to the URL (as in the “annotationsFile” parameter), such as user ID or document ID, in order to send user or document specific annotations. ViewONE will then tag on its own parameters to this URL so it is important to terminate the URL value correctly. That is, terminate the last parameter value with an ampersand (&), or if there are no parameters then terminate the value with the question mark (?). This will then conform to a standard URL format.

If you are unsure what ViewONE is sending through to the object then you can enable the annotations “debug” output (see page 43).

Note that the server-side object will only be called if the document displayed has been retrieved through a web server. For example, if a document is retrieved through the file:/// protocol then the server-side object will not be called when the user hits the annotations save button.

Parameter: **annotationAutoSave**

<PARAM NAME=“annotationAutoSave” value=“true”>

When this parameter is set to “true”, the user will not be prompted to save annotations. Instead, annotations will be sent automatically to the server-side save object. When the parameter is set to “false”, the user is prompted to save annotations.

The parameter applies when the applet is closed via a web page change, or when the document is closed via a document open or close action. It comes into effect regardless of whether you are using the “annotationSave”, “annotationSaveServlet” or “annotationSavePost” parameter.

The default value is “false”.

Parameter: **annotationAutoPrompt**

<PARAM NAME=“annotationAutoPrompt” value=“false”>

When this parameter is set to “false”, the user will not be prompted to save annotations nor will they be saved even if they require saving. This will override the “annotationAutoSave” parameter (if one was specified). When the parameter is set to “false” the only way annotation changes can be saved is when the user specifically clicks on the “save annotations” button (or the equivalent JavaScript method is called).

The parameter applies when the applet is closed via a web page change, or when the document is closed via a document open or close action. It comes into effect regardless of whether you are using the “annotationSave”, “annotationSaveServlet” or “annotationSavePost” parameter.

The default value is “true”.

Parameter: **annotationLocalSave**

<PARAM NAME="annotationLocalSave" value="false">

When this parameter is set to "true" and ViewONE is set to retrieve an image using the file:/// protocol, ViewONE will not save annotations using the specified server-side object. Instead, it will save an annotations definition file with the image.

When the parameter is set to "false" ViewONE will always save annotations by calling the specified server-side object.

The default value is "true".

Parameter: **annotationSuppressEmptyMessages**

<PARAM NAME="annotationSuppressEmptyMessages" value="true">

When this parameter is set to "true" ViewONE will not display an error message if the supplied annotations definition file is empty and there is no [EMPTY] marker. Further, it will allow the image to be displayed.

This parameter is to allow backwards compatibility with earlier ViewONE versions where empty annotations files were ignored.

The default value is "false" (always displays the error message and stops viewing).

Parameter: **annotationEncoding**

<PARAM NAME="annotationEncoding" value="UTF8">

This parameter forces a text encoding type that will allow multi-byte characters to be used. If however you machine is setup in a specific multi-byte character language (such as Chinese) then this parameter would not normally be required unless the machine is actually a non-Chinese machine and so only partially setup for Chinese support.

The value you must use to enable this feature is UTF8.

Important Note: When ViewONE send annotations then it will use this encoding and so the server object must make sure it can receive and preserve the encoding in the save and retrieval phases.

Early/old Java Engines: Early versions of Java did not support as many multi-byte characters as newer ones. We advise using Sun JRE 1.4.2+ for the widest support.

Parameter: **htmlEncoding**

<PARAM NAME="htmlEncoding" value="true">

This parameter applies at annotations save time. When it is set to "true" ViewONE will parse the annotations data and convert any of the following characters into their html encoded counterparts…

< > & "

…which are as follows…

< > & "

This parameter is useful if the server-side object that handles the annotations data stream will not accept non-html encoded data for security reasons.

The default value is "false".

Parameter: **AnnotationAutoSaveJ2Shutdown**

<PARAM NAME="annotationAutoSaveJ2Shutdown" value="true">

This parameter, when set to true, specifies that ViewONE should automatically save annotation changes when it detects that Sun Java Plugin version1.2 or later is detected to be in use by the browser. This behavior is required because the auto-save dialog of ViewONE can sometimes cause the browser to hang after the user has closed the browser window containing ViewONE under certain versions of the Sun Java Plugin.

The default value for this parameter is "false".

Mandatory parameters

Parameter: **annotate**

<PARAM NAME="annotate" value="false">

When this parameter is set to "false" all annotations features are disabled in the applet. Annotations cannot be viewed or edited, and are not retrieved.

The default value is "true".

Parameter: **annotateEdit**

<PARAM NAME="annotateEdit" value="true">

When this parameter is set to "true", the user may edit editable annotations and add new annotations via the annotations toolbar. When this parameter is set to "false", annotations cannot be edited (or added – only viewed), and the annotations toolbar is removed from the interface.

The default value is "false".

User interface defaults

Parameter: **hideAnnotationToolbar**

(V3-Only)

<PARAM NAME="hideAnnotationToolbar" value="true/false">

This parameter can be used to hide the annotation toolbar but maintain annotation edit features. Normally, if the toolbar is not visible then annotations cannot be created or edited. However there may be times when it is desired to hide the toolbar and still allow editing of annotation (perhaps to stop users creating new annotation but allow them to change existing ones).

In order to achieve this, set the AnnotateEdit HTML parameter (described in the previous section) to true, and set this parameter to false.

Parameter: **annotationHideButtons**

<PARAM NAME="annotationHideButtons" value="freehand, hyperlink, text">

This parameter specifies which buttons to hide on the annotations toolbar. Hiding a button only disables the ability to create an annotation through its user interface. Buttons are specified in a comma-delimited string using the following terms…

	restore	Restore button
	save	Save button
	select	Select mode button
	line	Line annotation
	arrow	Arrow annotation
	text	Text annotation
	solidText	Solid text annotation
	note	Note annotation
	highlight	Highlight annotation
	hyperlink	Hyperlink annotation
	highlightPoly	Polygon highlight annotation
	rectangle	Rectangle annotation
	Square	Square annotation **(V3-only)**
	redact	Redaction annotation
	redactPoly	Polygon redaction annotation
	poly	Polygon annotation
	openPoly	Open polygon annotation
	oval	Oval annotation

	circle	Circle annotation **(V3-only)**
	freehand	Freehand annotation
	stamp	Stamp annotation
	ruler	Ruler annotation **(V3-only)**
	angle	Angle annotation **(V3-only)**
	show	Show annotations

The terms are case insensitive.

If all terms are disabled the user will not be able to create annotations, only edit ones that already exist. The annotations toolbar will be disabled. The exception here is the show annotations button, which is located on the ViewONE status bar.

The default is to display all buttons.

Parameter:

annotationHideContextButtons

<PARAM NAME="annotationHideContextButtons" value="fillcolor, hyperlink">

This parameter specifies which buttons to hide on the context sensitive toolbar. Hiding a button only disables the ability to modify an annotation's default properties through the user interface. Buttons are specified in a comma-delimited string using the following terms…

	save	Fix button
	security	Edit security button
	text	Text editor button
	increaseline	Increase line width
	decreaseline	Decrease line width
	increasefont	Increase text font size
	decreasefont	Decrease text font size
	increasearrowhead	Increase arrowhead size
	decreasearrowhead	Decrease arrowhead size
	linecolor	Line color chooser
	fillcolor	Fill color chooser
	transparent	Make text semi-transparent
	hyperlink	Hyperlink dialog
	rotater	Rotate clockwise
	rotatel	Rotate counterclockwise
	angleflip	Flip angle (changes between obtuse & acute) **(V3-only)**
	behind	Move behind button
	delete	Delete button (delete key will still be enabled)

The terms are case insensitive.

If all terms are disabled the user will not see the context sensitive toolbar.

The default is to display all buttons.

Parameter: **annotationHideContextButtonsIds**

<PARAM NAME="annotationHideContextButtonsIds" value="note, text">

This parameter is used with the annotationHideContextButtons parameter to specify which annotation types to apply the annotationHideContextButtons values to.

For example if you specify "hyperlink" as an annotationHideContextButtons value and specify "note" as an annotationHideContextButtonsIds value, then the context sensitive hyperlink dialog button will be disabled for only the note annotation type. Annotation types are specified in a comma-delimited string using the following terms…

arrow, freehand, highlight, highlightpoly, line, note, openpoly, oval, poly, rectangle, redact, redactpoly, stamp, text

The terms are case insensitive.

The default is to apply the annotationHideContextButtons values to all annotation types.

Parameter: **annotationAllowHideAll**

<PARAM NAME="annotationAllowHideAll" value="true">

When the value of this parameter is "true", the user can use the "Hide Annotations" user-interface button to hide all annotations, including those that the user is not permitted to edit.

The default value is "false", which does not allow the user to hide annotations that it is not permitted to edit.

Parameter: **annotationEditButton**

<PARAM NAME="annotationEditButton" value="false">

When this parameter value is set to "false" it removes the Annotation Edit button from the top toolbar. The Annotation Edit button is used to hide and show the annotation toolbar.

The default value is "true".

Parameter: **annotationDblClick**

<PARAM NAME="annotationDblClick" value="true">

This parameter changes the default mouse-click operation for activating annotations. When set to "true" annotations will require a double left-click to activate them (note that this is different from selecting an annotation). This impacts sticky notes and annotation hyperlinks.

Note, this parameter only applies when the user interface is not in annotation edit mode - i.e., when the annotations toolbar is hidden or disabled.

The default value is "false" (single left-click).

Parameter: **annotationTabLength**

<PARAM NAME="annotationTabLength" value="4">

This parameter changes the default tab length (3) to the value specified. This length specifies the number of spaces to insert into a text annotation when the user presses the "TAB" key.

Parameter: **annotationLimitedColors**

<PARAM NAME="annotationLimitedColors" value="true">

This parameter changes the colors allowed for annotations within ViewONE. Specifically, it prevents the user from selecting white or near-white colors for annotations by modifying the color chooser dialogs.

It also prevents white or near-white colors from being used in annotation definitions files the viewer reads or writes. When one of these colors is found within a definitions file, the color is automatically darkened.

This makes it impossible to create redaction annotations with a white color.

The default value is "false".

Parameter: **annotationClearTextOnSelection**

<PARAM NAME="annotationClearTextOnSelection" value="true">

This parameter changes the behavior of text annotations when they are selected by the user. When the parameter is set to "true", a text annotation will be cleared automatically (i.e., it's text is cleared).

This feature may be useful for applications where users click on existing text annotations regularly with the intent of changing the entire text. This parameter will assist those users by removing the need to first select the text to delete it.

The "annotationClearTextList<*N*>" parameter can also be used to control this feature (see page 23).

The default value is "false".

Parameter: **annotationClearTextList<*N*>**

<PARAM NAME="annotationClearTextList1" value="Name expected here">
<PARAM NAME="annotationClearTextList2" value="Comments go here">

This parameter is used with the "annotationClearTextOnSelection" parameter (see page 23) so that only text annotations with certain content are cleared when they are selected by the user.

The particular content that the viewer looks for is specified in the value of each annotationClearTextList<*N*> instance. The value must match a text annotation's content exactly for the annotation to be cleared. There can be no leading or trailing characters.

Parameter: **prefOverrideAnnotationToolbar**

<PARAM NAME="prefOverrideAnnotationToolbar" value="true">

The parameter will override the user preference for showing the annotation toolbar.

Ordinarily the toolbar is hidden by default and after the user has shown the toolbar (by clicking on the "Show Annotation Toolbar" button) it is re-shown at the start of the user's next session.

Setting this parameter to "true" will force the toolbar to be visible for all sessions, irrespective of how the user last left it.

Parameter: **annotationsSticky**

<PARAM NAME="annotationsSticky" value="true">

This parameter keeps the cursor in annotations "drawing" mode and allows the re-use of a selected annotation type without having to re-select it from the annotations toolbar.

The default value for this parameter is "false".

annotationAutoWrap

Parameter: **annotationAutoWrap**

<PARAM NAME="annotationAutoWrap" value="false">

This parameter is used to stop text annotations wrapping at the end of the page. Since build 3.0.226 text annotations are automatically wrapped to multiple lines is the text spills over the edge of a page.

The default value for this parameter is "true".

Defaults for user generated annotations

Parameter: **userID**

<PARAM NAME="userID" value="user1">

This parameter specifies the value to identify a user for user-specific areas of ViewONE. Within the context of annotations this applies to annotation's "createdID" and "modifiedID" properties. When ViewONE sends new or modified annotations to the server, it will include (for each annotation) the createdID and modiedID properties set to the appropriate userID (see later descriptions of "createdID" and "modiedID" on page 69).

There is no default value.

Parameter: **annotationHighlightColor**

<PARAM NAME="annotationHighlightColor" value="0, 255, 0">

This parameter changes the default color for highlight annotations to the value specified. The value can be either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A.

The default values are 255, 255, 0 (yellow).

Parameter: **annotationLineColor**

<PARAM NAME="annotationLineColor" value="0, 0, 0">

This parameter changes the default color for lines to the value specified. The value can be either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A.

The default values are 255, 0, 0 (red).

Parameter: **annotationNoteColor**

<PARAM NAME="annotationNoteColor" value="255, 175, 175">

This parameter changes the default color for note annotations to the value specified. The value can be either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A.

The default values are 255, 255, 153 (pale yellow).

Parameter: **annotationNoteSize**

<PARAM NAME="annotationNoteSize" value="40, 55">

This parameter changes the default size for note annotations to the values specified. The value breaks down into width and height in that order (from left to right). The dimensions are in image pixels.

The default values are 50, 65.

Parameter: **annotationNoteRectangular**

<PARAM NAME="annotationNoteRectangular" value="true">

When this parameter is set to "true" the default style for note annotations is changed to rectangular (i.e., without leaf turn up).

The default value is "false".

Parameter: **annotationRedactColor**

<PARAM NAME="annotationRedactColor" value="255, 0, 0">

This parameter changes the default color for redaction annotations to the value specified. The value can be either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A.

The default values are 0, 0, 0 (black).

Parameter: **annAnnotationFont**

<PARAM NAME="annAnnotationFont" value="courier">

This parameter changes the default font for text annotations. The value should a supported font type, currently only "arial" and "courier" are supported by ViewONE. If an unsupported font type is specified, the default will be assumed.

The default for this parameter is "arial".

Parameter: **annotationTextColor**

<PARAM NAME="annotationTextColor" value="0, 0, 0">

This parameter changes the default color for text to the value specified. The value can be either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A.

The default values are 255, 0, 0 (red).

Parameter: **annotationTextFillColor**

<PARAM NAME="annotationTextFillColor" value="255, 255, 255">

This parameter changes the default fill color for filed text annotations to the value specified. The value can be either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A.

The default values are 255, 255, 153 (pale yellow).

Parameter: **allowTextRubberband**

<PARAM NAME="allowTextRubberband" value="false">

(Added build 252, 21/Jan/2005). This parameter stops ViewONE from allowing text annotations to be resized (the bounding box – also called a rubberband box is disabled).

The default value is True.

Parameter: **annotationHyperlinkWeb**

<PARAM NAME="annotationHyperlinkWeb" value="http://www.mysite.com">

This parameter changes the default base location for web hyperlinks from the codebase path to the URL specified. Relative hyperlinks are then relative to that URL. This makes it easier for annotations users to specify hyperlinks for their annotations.

For example, if we set the parameter as in the above case then we specify an annotation's web hyperlink to be "page.html", the actual hyperlink path would resolve to "http://www.mysite.com/page.html".

The default value is the codebase path.

Parameter: **annotationStamp<*N*>**

<PARAM NAME="annotationStamp1" value="image:logo.tif">
<PARAM NAME="annotationStamp2" value="Today's date is <date>">

This parameter specifies the stamps to use in place of ViewONE's default stamps (see Appendix B). The stamp specified can be either text or image based.

An image stamp is an overlaid image which may be in any of the image formats supported by ViewONE. An image-based stamp is recognized by the use of the "image:" identifier. The path that follows the identifier specifies the location of the annotations definition file relative to the applet's codebase.

A text-based stamp may also contain several optional identifiers which are substituted for actual values at runtime. They are…

Identifier	*Description*
<date>	Inserts the date, time and time zone from the user's machine at the time the stamp is placed.
<dateonly>	Inserts the date from the user's machine at the time the stamp is placed.
<userid>	Inserts the userID as specified by the viewer's "userID" parameter.

Additional properties can be specified using the "annotationStampProperties<*N*>" parameter, such as a transparent color for image stamps, or color and size for text stamps. You can change the date style for the <date> and <dateonly> identifiers through the "annotationDateStyle" parameter.

ViewONE's default stamps are listed in Appendix B.

Parameter: **annotationStampProperties<*N*>**

<PARAM NAME="annotationStampProperties1" value="<menu=Company logo><color=white>">
<PARAM NAME="annotationStampProperties2" value="<menu=Today's Date><color=black><rotation=45>">

This parameter specifies various properties for stamps to use in place of ViewONE's default stamps (see Appendix B). The stamp properties specified can be either text or image based.

The parameter is for use only when the "annotationStamp<*N*>" parameter is used. Each use of the parameter corresponds to an instance of the "annotationStamp<*N*>" parameter. For example, "annotationStampProprties1" corresponds to "annotationStamp1", and so on. If the parameter is used at all then all stamp properties for the stamps menu must be specified. For example, you cannot have "annotationStampProperties2", but not "annotationStampProperties1".

The format of the properties string is as follows:

“<property1=value1><property2=value2>…

e.g “<color=blue>”, or “<color=blue><rotation=45>”

The optional properties are as follows…

Property name	Description and value options
Color	Color of text or line (or transparency color for image stamps)
FillColor	Background Color of text or fill color of shape
Menu	Menu text to display (for stamp menus)
Rotation	Rotation of annotation (0, 45, 90, 180 or 270)
Label	Text for the annotation label
HelpText	Tooltip text
Owner	Owner (username) text
FontHeight	Height in pixels for text/stamp annotations
Print	True = other users can print annotation False = other users cannot print annotation
Read	True = other users can read annotation False = other users cannot read annotation
Modify	True = other users can modify annotation False = other users cannot modify annotation
Delete	True = other users can delete annotation False = other users cannot delete annotation
ModifySecurity	(for annotation security model 2 – see page 41) True = other users can modify annotation security False = other users cannot modify annotation security

ViewONE’s default stamps are listed in Appendix B.

Parameter: **annotationDateStyle**

<PARAM NAME="annotationDateStyle" value="short">

This parameter specifies the date style used for the "annotationStamp" parameter's <date> and <dateonly> identifiers. It also alters the date style reported in the sticky note dialog.

There are two formats available, "long" and "short". The actual format is determined by the client machine's regional settings, but examples of the two formats are…

Long: 24 September, 2002, 13:00:01, GMT+01:00
Short: 24-Sep-02, 13:00:01, GMT+01:00

The default value is "long".

Parameter: **annotationTarget<*N*>**

<PARAM NAME="annotationTarget1" value="<_self><This window>">
<PARAM NAME="annotationTarget2" value="<_blank><New window>">
<PARAM NAME="annotationTarget3" value="<banner><Banner frame>">

This parameter specifies a list of optional targets for web hyperlinks to use in place of ViewONE's default targets (see Appendix C). The value specified follows the format…

<target><menu label text>

This is where "target" can be any of the following…

Target	***Description of target***
_self	Current web page
_blank	New window
_parent	Parent frame
_top	Top most parent frame
Any frame window name	This window must exist for a successful link

This is useful for specifying common windows in a web application, for example, index terms, contents, help, etc.

If the parameter is used at all then all hyperlink targets must be specified. For example, you cannot have "annotationTarget2", but not "annotationTarget1".

ViewONE's default targets are listed in Appendix C.

Parameter: **annotationFreehandLimit**

<PARAM NAME="annotationFreehandLimit" value="20">

This parameter specifies the maximum number of points for a freehand annotation. When this limit is reached during the drawing of a freehand annotation, the annotation drawing is automatically stopped.

The default value for this parameter is 'unlimited' points. The value of this parameter must be greater than 1.

Parameter: **defaultfFontHeight**

<PARAM NAME="defaultFontHeight" value="20">

This parameter specifies the default font height for new text annotations. ViewONE creates new text annotations with a font height of 28 pixels for image documents and 48 for text documents and this parameter can be used to adjust these values. Both values are set to the specified value.

Parameter: **defaultLineWidth**

(v3-Only)

<PARAM NAME="defaultLineWidth" value="5">

This parameter specifies the default line width for new. ViewONE creates new annotations with a line width of 8 pixels and this parameter can be used to adjust this value.

Parameter: **annotationDefaultLineColor*N***

<param name="AnnotationDefaultLineColor1 " value="line: 255, 0, 0">

<param name="AnnotationDefaultLineColor2 " value="arrow: 0, 0, 255">

This parameter specifies the default line color for a given annotation type. The example above will change the default color of a line annotation to red and the default color of an arrow annotation to blue.

To set the default color of a specific "Stamp" annotation, simply insert part of the "menu" name rather than the annotation type. For example, to change the default color for the standard ViewONE stamp "Approved" simply insert the following into the html:

<param name="AnnotationDefaultLineColor1 " value="approved: 255, 0, 0">

Parameter: **annotationDefaultFillColor*N***

<param name="AnnotationDefaultFillColor1 " value="highlight: 255, 0, 0">

<param name="AnnotationDefaultFillColor2 " value="oval: 0, 0, 255">

This parameter specifies the default fill color for a given annotation type. The example above will change the default fill color of a highlight annotation to red and the default fill color of an oval annotation to blue.

Parameter: **unitDecimalPlaces**

(V3-Only)

<param name="UnitDecimalPlaces" value="5">

This parameter specifies the number of decimal places the ruler and angle annotations displaying measurements. The default is 3 decimal places.

Parameter: **rulerUnits**

(V3-Only)

<param name="RulerUnits" value="cm">

This parameter specifies the units the ruler annotation displays. The optional values are:

cm	*centimeters*
mm	*millimeters*
inches	*inches*
inchesandcm	*inches and centimeters*

Parameter: **rulerScale**

(V3-Only)

<param name="RulerScale" value="2.0">

This parameter specifies the scale applied to the ruler units. The scale is basically a multiplier, so a scale of 2.0 means the figures the ruler displays will be multiplied by 2. The default scale is 1.0.

Parameter: **angleUnits**

(V3-Only)

<param name="AngleUnits" value="rad">

This parameter specifies the units the angle annotation displays. The optional values are:

degrees	*centimeters*
rad	*radians*
dms	*degrees, minutes and seconds*

Using annotations as headers and footers

This feature allows an additional annotations definition file to be specified specifically for use as a print header and footer. Each annotation in the file is repeated on every page of the document. Users cannot edit these annotations.

You must define a separate annotation in the additional annotations definition file for each header or footer you wish to add to each page of the document.

Below is an example additional annotation definition file containing a single header and a single footer:

```
[TEXT]
FONTTYPE = arial
FONTHEIGHT = 40
SEMITRANSPARENT = 0
BORDER = 0
TEXT = This is the header
X = 10
Y = 10
PAGE = -1
TRANSPARENT = 1
LABEL = Text1
PAGESIZE = 1728, 2175
EDIT = 0
CREATEDATE = 17 Feb 2004, 14:13:00, EST
MODIFIEDDATE = 17 Feb 2004, 14:13:00, EST

[TEXT]
FONTTYPE = arial
FONTHEIGHT = 40
SEMITRANSPARENT = 0
BORDER = 0
TEXT = This is the footer
X = 10
Y = 2100
PAGE = -1
TRANSPARENT = 1
LABEL = Text1
PAGESIZE = 1728, 2175
EDIT = 0
CREATEDATE = 17 Feb 2004, 14:13:00, EST
MODIFIEDDATE = 17 Feb 2004, 14:13:00, EST
```

When this additional annotations definition file is used in conjunction with the “annotationTemplate” parameter (which is used to specify the location of the additional annotations definition file), a header and footer will be added to each page of the document at the coordinates defined by the X and Y parameters.

It is possible to add page specific information to the annotations supplied by the annotations definition file using the "annotationSubstitution<*N*>" or "annotationTemplateValuesFile" parameters in conjunction with markers defined in the annotation definitions. For more information and an examples of using substitutions, see the annotationSubstitution<*N*> section below.

Header and/or footer annotations are not limited to just text annotations, any type of annotation can be used. Therefore, it is possible to add watermarks, logos or even notes to each page of the document using this method.

Parameter: **annotationTemplate**

<PARAM NAME="annotationTemplate" value="../templates/t1.ant">
<PARAM NAME="annotationTemplate" value="http://www.mysite.com/myscript.pl?tmpID=8791">

This parameter provides a URL to an annotations file to be used as a print header or footer. It specifies the location of the annotations definition file relative to the applet's codebase. The value can also specify a server-side object (CGI, EXE, ASP, etc.) that streams an annotations file to the applet.

Parameters may be added to the URL, such as user ID or document ID, in order to provide user or document specific annotations. If you use a server-side object to supply/stream this file then it is up to this object to handle any parameters you may include.

The annotations file itself must conform to the ViewONE annotations file specification described later in this manual (see page 65). There are, however a few extra rules that apply:

1. Template definition annotations should always have a PAGE value of -1.

2. Annotation substitutions may be used within the annotations file (see annotationSubstitution<*N*> parameter below).

If you are unsure whether ViewONE has processed your annotations file correctly then you can enable the annotations "debug" output (see page 43).

Parameter: **annotationSubstitution<*N*>**

<PARAM NAME="annotationSubstitution1" value="1: <document>=my document, <page>=1">
<PARAM NAME="annotationSubstitution2" value="2: <page>=2, <id>=AGH13534">

This parameter specifies a list of tags that match up to corresponding tags within TEXT type annotations from the annotations definition data, which in turn is specified by the "annotationTemplate" parameter. Within the value part of the parameter, values are associated with those tags.

When the template annotations file is downloaded and interpreted those tags are replaced with the values associated with them through this parameter.

The number of annotationSubstitution parameters specified is not limited to the number of pages in the document, only the number of tags in the annotations template file. Each instance of an annotationSubstitution parameter can only contain tags and values for one page at a time.

NOTE: The value String must only contain a colon(:) after the page marker. No other colons are allowed in the value string. Also, the only comma(,) characters allowed, are those separating the substitution values, you cannot use commas in your substitution text.

The value string can be broken down in the following way…

page_number: <*tag*>=*associated_value*[, <*tag*>=*associated_value*]

For example, a value string of…

1: <document>=my document, <page>=1

…means: look at the text annotations on page one of the document and replace any instances of the <document> tag with "my document" and any instances of the <page> tag with "1".

A value string of…

2: <page>=2, <id>=AGH13534

…means: look at the text annotations on page two of the document and replace any instances of the <page> tag with "2" and any instances of the <id> tag with "AGH13534".

There are two special tags <DATE> and <DATEONLY> that when used as a tag in the annotation definition will be substituted with the system date of the local machine. If <DATEONLY> is used then just the date is substituted for the tag. If the <DATE> tag is used then the current date and time are used. The format of the date string can be changed using the annotationDateStyle parameter (see page 31).

A special value for substituting tags with empty strings must be used if you want to remove a tag contained in an annotation. The value is <EMPTY> as follows…

1: <document>=<EMPTY>

This will clear any <document> tags in the annotation.

A full example can be found in the "example annotation file and HTML" section below.

Parameter: **annotationTemplateValuesFile**

<PARAM NAME="annotationTemplateValuesFile" value="../templates/t1.txt">
<PARAM NAME="annotationTemplateValuesFile" value="http://www.mysite.com/myscript.pl?tmpID=8791">

This parameter is an alternative to the "annotationSubstitution<*N*>" parameter. It points ViewONE to a text file that lists the "annotationTemplate" text annotation tags and their associated values.

That location is specified through a URL, relative to the applet's codebase. If desired, the URL can specify a server-side object (CGI, EXE, ASP, etc.) that streams a template values file to the applet.

Parameters may be added to the URL, such as user ID or document ID, in order to provide user or document specific template values. If you use a server-side object to supply/stream this file then it is up to this object to handle any parameters you may include.

Inside the txt file there is no need for the parameter name, just the string values. They break down in the same way as with the annotationSubstitution<*N*> parameter, i.e…

page_number: <*tag*>=*associated_value*[, <*tag*>=*associated_value*]

For example, a value string of…

1: <document>=my document, <page>=1

…means: look at the text annotations on page one of the document and replace any instances of the <document> tag with "my document" and any instances of the <page> tag with "1".

A value string of…

2: <page>=2, <id>=AGH13534

…means: look at the text annotations on page two of the document and replace any instances of the <page> tag with "2" and any instances of the <id> tag with "AGH13534".

The values are listed one per line in the text file. A carriage-return, a linefeed, or a carriage-return and linefeed then delimit the end of the line. Like this…

1: <document>=my document, <page>=1
2: <page>=2, <id>=AGH13534

…and so on

Example Annotation File and HTML

An additional annotations definition file, headerandfooter.txt containing the following:

```
[TEXT]
FONTTYPE = arial
FONTHEIGHT = 40
SEMITRANSPARENT = 0
BORDER = 0
TEXT = <HEADER> Page: <PAGE> DOC: 1234356 <DATE>
X = 10
Y = 10
PAGE = -1
TRANSPARENT = 1
LABEL = Text1
PAGESIZE = 1728, 2175
EDIT = 0
CREATEDATE = 17 Feb 2004, 14:13:00, EST
MODIFIEDDATE = 17 Feb 2004, 14:13:00, EST

[TEXT]
FONTTYPE = arial
FONTHEIGHT = 40
SEMITRANSPARENT = 0
BORDER = 0
TEXT = <FOOTER> Page: <PAGE> DOC: 1234356 <DATE>
X = 10
Y = 2100
PAGE = -1
TRANSPARENT = 1
LABEL = Text1
PAGESIZE = 1728, 2175
EDIT = 0
CREATEDATE = 17 Feb 2004, 14:13:00, EST
MODIFIEDDATE = 17 Feb 2004, 14:13:00, EST
```

When used in conjunction with the following HTML parameters:

```
<PARAM NAME="annotationTemplate" value="./headerandfooter.txt">

<PARAM NAME="annotationSubstitution1" value="1:<HEADER>=header for page 1 -,
<FOOTER>=footer for page 1 -,, <PAGE>=1">

<PARAM NAME="annotationSubstitution2" value="2:<HEADER>=A different header -,
<FOOTER>=A different footer -, <PAGE>=2">
```

<PARAM NAME="annotationSubstitution3" value="3:<HEADER>=<EMPTY>,
<FOOTER>=<EMPTY>, <PAGE>=2">

Would result in the following header and footer on the first page of the document:

“header for page 1 – Page: 1 DOC: 123456 1st April 2004”
“footer for page 1 – Page: 1 DOC: 123456 1st April 2004”

The following header and footer on the second page of the document::

“A different header – Page: 2 DOC: 123456 1st April 2004”
“A different footer – Page: 2 DOC: 123456 1st April 2004”

And the following header and footer on the third page of the document::

“Page: 2 DOC: 123456 1st April 2004”
“Page: 2 DOC: 123456 1st April 2004”

Security

Parameter: **annotationJavascriptExtensions**

(V3-Only)

<PARAM NAME="annotationJavascriptExtensions" value="true">

This parameter controls the use of some of the annotations related JavaScript methods. This provides an extra level of security when ViewONE is in an environment where JavaScript calls can be issued to it. That is, it allows control over whether the viewer should or should not accept calls for those methods.

The methods this parameter has control over are…

- reloadAnnotations()
- setAnnotateEdit(*boolean*)
- int getNumAnnotations(*type*, *page*)
- string getAnnotationLabels(*type*, *page*)
- string getAnnotationLabels(*label*)
- addAnnotation(*annotationProperties*)
- modifyAnnotation(*label*, *annotationProperties*)
- deleteAnnotation(*label*)
- deleteAllAnnotations(*type*, *page*)
- startAnnotation(*type*)

The default value for this parameter is "false".

Parameter: **annotationSecurityModel**

<PARAM NAME="annotationSecurityModel" value="2">

This parameter sets the security model to be used for annotation security. The values available are as follows:

1 – Simple Annotation Security (see page 62 for details)

2 – Extended Annotation Security (see page 63 for details)

The default value for this parameter is "1".

Parameter: **userAdmin**

<PARAM NAME="userAdmin" value="true">

This parameter, when set to true, specifies the current user as being an administrator, with the ability to edit security settings for any annotation.

The default value for this parameter is "false".

Parameter: **annotationEditPasswordModify**

<PARAM NAME="annotationEditPasswordModify" value="true">

When this parameter is set to "true" a further option is presented in the security dialog when using extended annotation security (see page 63).

This dialog will allow the user to set a password for modifying annotations. Only the owner or an admin user will be able to change these passwords once created. The default value for this parameter is "false".

Parameter: **annotationEditPasswordSecurity**

<PARAM NAME="annotationEditPasswordSecurity" value="true">

When this parameter is set to "true" a further option is presented in the security dialog when using extended annotation security (see page 63).

This dialog will allow the user to set a password for setting the security for annotations. Only the owner or an admin user will be able to change these passwords once created. The default value for this parameter is "false".

Parameter: **annotationEditPasswordText**

<PARAM NAME="annotationEditPasswordText" value="true">

When this parameter is set to "true" a further option is presented in the security dialog when using extended annotation security (see page 63).

This dialog will allow the user to set a password for changing the text in a text annotation. Only the owner or an admin user will be able to change these passwords once created. The default value for this parameter is "false".

Miscellaneous

Parameter: **annotationStart**

<PARAM NAME="annotationStart" value="index1">

This parameter defines a particular annotation which ViewONE will ensure is displayed when a document is opened. The value specified is the label of an annotation on the newly opened document. It is not case sensitive.

If the annotation is on a page other than the first page, then ViewONE will automatically change to the required page. ViewONE will also apply any windows settings which may have been saved with that annotation, such as zoom factor, scrolling, rotation etc. If no settings have been saved with that annotation ViewONE will attempt to scroll the page so that the annotation is in the top left region of the image display area using the current scale setting (fit-to-width/height etc).

There is no default value.

Note that any annotation can have "window settings". This a feature unique to ViewONE that allows the user to easily setup a hyperlink (or start annotation) so that when that annotation is "activated", by clicking on a hyperlink to that annotation (or by using the "annotationStart" parameter), ViewONE can be made to show an annotation or a specific part of an image at any chosen zoom factor, scroll position, etc. (see hyperlink section for further information – page 71).

Parameter: **annotationTrace**

<PARAM NAME="annotationTrace" value="true">

When this parameter is set to "true", ViewONE will output a trace to the Java Console to display annotation file processing. This may be useful for checking whether your annotations file has been processed or defined correctly, and to locate general problems when saving annotations.

To see this trace you will need to enable the Java Console (if you have not already done so) and have it visible while viewing annotations.

The default value for this parameter is "false".

Note 1: The trace will degrade performance as a result of having to output the additional information to the console (some Java Consoles perform better than others). For this reason it is best only to enable this parameter when debugging.

Note 2: When "annotationTrace" is set to "false" the applet will still output some error messages to the Java Console to assist you when problems are detected.

Note 3: To enable the Java console…

…for Internet Explorer (PCs) :

1) Start your browser, go to the "View" menu at the top of the window and see if you have the "Java Console" option available. If you do, then select it to bring up the console. If you do not have it available then continue to 2).
2) If you are using Internet Explorer 4.x on the PC then go to the "View" menu again and "Internet Options". If you are using Internet Explorer 5.x on the PC then go to the "Tools" menu and select "Internet Options".
3) Select the "Advanced" tab - it's located near the top right of the box that has popped up.
4) Scroll down the list that has appeared until you get to the "Java VM" or "Microsoft VM" sections. Then check the "Java console enabled" box and click "OK". Then close your browser.
5) Start your browser, go to the "View" menu and select "Java Console".

…for Netscape Communicator 4.x :

1) Start your browser, go to the "Communicator" menu at the top of the window and select "Java Console".

…for Netscape 6.0 (PCs) :

1) Go to your machine's control panel and double-click on the "Java Plug-in 1.3" icon.
2) The Java Plug-in Control Panel should then appear. Check the "Show Java Console" box, hit "Apply" then close the panel. Now when you start your browser the Java Console should also come up in the background.

…for Netscape 6.0 (Linux/Unix) :

See Sun's Java Plug-in documentation, available from their site at http://java.sun.com.

Parameter:

renderSupport

<PARAM NAME="renderSupport" value="true">

When this parameter is set to "true", ViewONE will attempt to use more system resources to improve performance. The areas of the viewer affected are scrolling, dragging, zooming and printing.

The default value for this parameter is "false".

Note: This parameter should only be set to true for use with Internet Explorer.

Parameter: **annotationCache**

<PARAM NAME="annotationCache" value="false">

When this parameter is set to "false", ViewONE will bypass the browser's caching when retrieving annotations definition files. This may be needed where annotations definition data is created on demand using a server-side streaming object.

When this parameter is set to "true", ViewONE will retrieve the annotations definitions file as though it never changes for a given image. This means that setting the parameter to true will help reduce network and server overhead because the browser will first use its cache before going to the server for the file.

The default value for this parameter is "true".

Parameter: **annotationColorMask**

<PARAM NAME="annotationColorMask" value="abgr">

This parameter defines how the colors are made up when specifying "long" values for annotation colors. It specified the location of the color elements in the string value. The color elements are…

alpha	a
red	r
green	g
blue	b

In the above example the four bytes are ordered alpha, blue, green and then red. Red is the low order part of the long, alpha is the high order part.

If an element of the value is not used then a 0 should be used in its place.

The default value is "0rgb".

Parameter: **adjustFontScale**

<PARAM NAME="adjustFontScale" value="0.931">

This parameter permits the size of characters used by ViewONE (for displaying text annotations and text files) to be fine tuned.

Since build 2.1.6296 ViewONE has included higher resolution fonts and slightly different character sizes to cater for extended language support (Arabic, Russian, Chinese and Japanese).

As a result, and to maintain matching font sizes for previous versions of ViewONE, this parameter may also be used to make the necessary adjustment to maintain backwards compatibility. This would ordinarily only be required where annotations have been overlaid on forms using earlier versions of ViewONE and where the forms require precise and exact similar positioning. In such cases a value of "0.931" will be required to maintain compatibility with previous versions of ViewONE.

The reader should be advised, however, that the new font capabilities also produce more accurate representation of the desired font size (please see the TEXT annotation's FONTHEIGHT annotation property). Previous versions of ViewONE may have misrepresented font sizes by a few pixels.

Wang Annotation Support

Parameter: **localAnnotationReadMode**

<PARAM NAME="localAnnotationReadMode" value="ANTnWANG">

Annotation reading mode for local files, i.e. images specified using a local or network path or opened via the file open dialog.

Can be specified using an integer or string as follows:

0. ("ANT") - Read ViewONE annotations as normal. (default)
1. ("WANG") - Read Wang annotations only from tif files
2. ("EMBEDDED") - Read ViewONE annotations embedded in tif file (not implemented yet)
3. ("ANTnWANG") – Read Wang annotations only from tif files and also normal ViewONE annotations from ant file.

The default value for this parameter is 0 – ANT

Parameter: **localAnnotationWriteMode**

<PARAM NAME="localAnnotationWriteMode" value="1">

Annotation writing mode for local files, i.e. images specified using a local or network path or opened via the file open dialog.

Can be specified using an integer or string as follows:

0. ("ANT") - Read ViewONE annotations as normal. (default)
1. ("WANG") – Write Wang annotations into the tif file. The image can be saved using a server side component if required (see ImageSave, ImageSaveServlet and ImageSavePost in the ViewONE HTML manual).
2. ("EMBEDDED") - Embed ViewONE annotations in the tif file (not implemented yet).
3. ("ANTnWANG") – Write Wang annotations into the tif image and any annotations unsupported by Wang (see page109) using the standard ViewONE save mechanism.

The default value for this parameter is 3 - ANTnWANG

Parameter: **unsupportedWangError**

<PARAM NAME="unsupportedWangError" value="false">

Tells ViewONE whether to put up an error message if saving annotations as Wang and unsupported types of annotation are found.

The default value for this parameter is “true”.

Parameter: **webAnnotationReadMode**

<PARAM NAME="webAnnotationReadMode" value="1">

Annotation reading mode for images from the server, i.e. images specified as a URL.

Can be specified using an integer or string as follows:

0. (“ANT”) - Read ViewONE annotations as normal. (default)
1. (“WANG”) - Read Wang annotations only from tif files
2. (“EMBEDDED”) - Read ViewONE annotations embedded in tif file (not implemented yet)
3. (“ANTnWANG”) – Read Wang annotations only from tif files and also normal ViewONE annotations from ant file.

The default value for this parameter is 0 - ANT

Parameter: **webAnnotationWriteMode**

<PARAM NAME="webAnnotationWriteMode" value="1">

Annotation writing mode for images from the server, i.e. images specified as a URL.

Can be specified using an integer or string as follows:

0. (“ANT”) - Write ViewONE annotations as normal. (default)
1. (“WANG”) - Write Wang annotations into the tif image. One of the save image parameters must be set so the new image, with Wang annotations, can be sent back to the server (see ImageSave, ImageSaveServlet and ImageSavePost in the ViewONE HTML manual).
2. (“EMBEDDED”) - Write ViewONE annotations into the tif image file. (not implemented yet)
3. (“ANTnWANG”) – Write Wang annotations into the tif image and any annotations unsupported by Wang (see page109) using the standard ViewONE save mechanism.

The default value for this parameter is 0 - ANT

Applet JavaScript

The JavaScript examples in this manual do not refer to their use in any particular context. The examples could be used within functions of a JavaScript program or directly as event handlers to buttons, hyper-links etc. Our web site illustrates such uses; alternatively refer to an appropriate JavaScript guide.

Filenames and hyperlink addresses are expressed using the Internet URL address format (Uniform Resource Locator), e.g. “http://mysite/myimage.tif”. If any part of the address before “myimage.tif” is not included then the applet will assume a base address that is the same as the applet location (the codebase).

With the exception of filenames and hyperlink addresses, all parameters are case insensitive.

Opening and saving annotations

Method:

setAnnotationFile(*path*)

ViewONE.setAnnotationFile("../docs/p1.ant");
ViewONE.setAnnotationFile("http://www.mysite.com/myscript.pl?userID=12&docID=8791");

ViewONE requires an annotations definition file which contains a list of the annotations to display (and their associated parameters). This parameter defines that file.

It specifies the location of the annotations definition file relative to the applet's codebase. The value can also specify a server-side object (CGI, EXE, ASP, etc.) that streams an annotations file to the applet.

Parameters may be added to the URL, such as user ID or document ID, in order to provide user or document specific annotations. If you use a server-side object to supply/stream this file then it is up to this object to handle any parameters you may include here.

The annotations file itself must conform to the ViewONE annotations file specification described later in this manual (see page 65).

If you are unsure whether ViewONE has processed your annotations file correctly then you can enable the annotations "debug" output (see page 43).

Note: To have an effect this method must be called before the document is opened or the reloadAnnotations() method (below) must be called following this method.

Method:

setAnnotationSavePost(*path*)

ViewONE.setAnnotationSavePost("http://www.mysite.com/annotationsave.dll");

Specifies the location of a server-side object (CGI, EXE, ASP, JSP, etc.) that handles the saving of annotations created by the viewer. It is up to whoever implements the system to supply the object, which must conform to the specification for ViewONE annotations save POST objects (see page 93).

If a relative address is used, the location it defines is relative to the applet's codebase.

If you are unsure what ViewONE is sending through to the object then you can enable the annotations "debug" output (see page 43).

Note 1: The object will only be called if the document displayed has been retrieved through a web server. For example, if a document is retrieved through the file:/// protocol then the object will not be called when the user hits the annotations save button.

Note 2: To have an effect this method must be called before the document is opened.

Method:

setAnnotationPostPrefix(*parameters*)

ViewONE.setAnnotationPostPrefix("p1=value1&p2=value2");

This method specifies the parameters to be added to the POST data for the annotationSavePost feature (see page 12). The above example demonstrates prefixing "p1=value1" and "p2=value2" to the POST data.

The POST object should then be able to parse the data posted to extract all the parameters, in this case p1 and p2, plus the usual size, numdata, data01, data02, etc. (see page 93).

Method:

setAnnotationSaveServlet(*path*)

```
ViewONE.setAnnotationSaveServlet(
        ”http://www.mysite.com/servlet/annotationsave.class?userID=12&docID=8791”);
```

Specifies the location of a servlet that handles the saving of annotations created by the viewer. It is up to whoever implements the system to supply the servlet, which must conform to the specification for ViewONE annotations save servlets (see page 94).

If a relative address is used, the location it defines is relative to the applet's codebase.

Parameters may be added to the URL (as in the “setAnnotationsFile” parameter), such as user ID or document ID, in order to send user or document specific annotations.

If you are unsure what ViewONE is sending through to the servlet then you can enable the annotations “debug” output (see page 43).

Note 1: The servlet will only be called if the document displayed has been retrieved through a web server. For example, if a document is retrieved through the file:/// protocol then the servlet will not be called when the user hits the annotations save button.

Note 2: To have an effect this method must be called before the document is opened.

Method:

setAnnotationSave(*path*)

ViewONE.setAnnotationSave("http://www.mysite.com/myscript.pl?userID=12&docID=8791&");

Specifies the location of a server-side object (CGI, EXE, ASP, JSP, etc.) that handles the saving of annotations created by the viewer using multiple HTTP:GET calls. It is up to whoever implements the system to supply the object, which must conform to the specification for ViewONE annotations save GET objects (see page 92).

If a relative address is used, the location it defines is relative to the applet's codebase.

Parameters may be added to the URL (as in the "setAnnotationsFile" method), such as user ID or document ID, in order to send user or document specific annotations. ViewONE will then tag on its own parameters to this URL so it is important to terminate the URL value correctly. For example, terminate the last parameter value with an ampersand (&), or if there are no parameters then terminate the value with the question mark (?). This will then conform to a standard URL format.

If you are unsure what ViewONE is sending through to the object then you can enable the annotations "debug" output (see page 43).

Note 1: The server-side object will only be called if the document displayed has been retrieved through a web server. For example, if a document is retrieved through the file:/// protocol then the server-side object will not be called when the user hits the annotations save button.

Note 2: To have an effect this method must be called before the document is opened.

Method:

saveAnnotations()

ViewONE.saveAnnotations();

Immediately forces ViewONE to save annotations with whatever server-side annotations save object has been specified (via the relevant HTML parameters or JavaScript methods).

Method: **reloadAnnotations()**

Forces ViewONE to reload the annotations from source and redraw the image on screen. This allows you to call the “setAnnotationFile” method without having to close down the document first.

For example…

```
ViewONE.setAnnotationFile(“http://www.mysite.com/myscript.pl?userID=12&docID=8791”);
ViewONE.reloadAnnotations();
```

V3 security note: This method is disabled by default unless the “annotationJavascriptExtensions” parameter is set to “true”.

Editing and finding annotations

Method: **setAnnotateEdit(*boolean*)**

ViewONE.setAnnotateEdit(true);

When this method is called with "true", the user may edit editable annotations and add new annotations via the annotations toolbar. When the method is called with "false", annotations cannot be edited (or added – only viewed), and the annotations toolbar is removed from the interface.

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **isAnnotateEdit()**

var boolAnnotateEdit = ViewONE.isAnnotateEdit();

Returns a boolean value of "true" if annotateEdit is enabled. Otherwise a value of "false" is returned.

Method: **isAnnotationsUpdated()**

var boolAnnotateUpdated = ViewONE.isAnnotationsUpdated();

Returns a boolean value of "true" if annotations have not been changed since they were last updated, i.e. saved in ViewONE. Otherwise a value of "false" is returned.

Method: **showAnnotationToolbar(*boolean*)**

ViewONE.showAnnotationToolbar(false);

Specifies whether to show or hide the main annotations toolbar.

Method: **isAnnotationToolbar()**

var boolAnnotationToolbar = ViewONE.isAnnotationToolbar();

Returns a boolean value of "true" if the main annotations toolbar is shown. Otherwise a value of "false" is returned.

Method: **getNumAnnotations(*type*, *page*)**

var intNumAnnotations = ViewONE.getNumAnnotations("highlight", 1);

Returns the number of annotations of the type specified on the page specified. The accepted *type* values are…

any
line
arrow
text
note
highlight
highlightPoly
rectangle
redact
redactPoly
poly
openPoly
oval
freehand
stamp

If a value of "any" is used then all types of annotation are included. If you specify a *page* value of –1 then all pages are included.

Notes ***(V3-Only):***

A circle annotation is actually an oval with fixed aspect ratio.
A square is actually a rectangle annotation with fixed aspect ratio.
A ruler is actually a line annotation with different display properties.
A angle is actually a openPoly annotation with different display properties.

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **getAnnotationLabels(*type*, *page*, *order*)**

var strAnnotationLabels = ViewONE.getAnnotationLabels("highlight", 1, 0);

Returns the labels of annotations of the type specified on the page specified in a delimited list.

The list is ordered either by annotation creation order or by X/Y position, depending on how the *order* parameter is set. If the parameter value is 0 then annotations are listed in the order they were added to the document (a last-in-first-out order). If the parameter value is 1 then annotations are listed in the order they are shown on the page (an X/Y position order).

The convenience method, getDelimiter() can be called to identify the delimiter used. The *type* and *page* parameter values have the same rules as in the getNumAnnotations(*type*, *page*) method (see page 56).

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **getAnnotation(*label*)**

var strAnnotationProperties = ViewONE.getAnnotation("myHighlight1");

Returns the properties of the annotation that has a label property matching the one specified. The properties are returned in a delimited list. The convenience method, getDelimiter() can be called to identify the delimiter used.

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **addAnnotation(*annotationProperties*)**

```
ViewONE.addAnnotation(
    "[HIGHLIGHT]<P>PAGE=0<P>X=300<P>Y=350<P>WIDTH=400<P>HEIGHT=500<P>
    LABEL=myHighlight1<P>HYPERLINK=<page><3><P>");
```

Adds an annotation with the specified properties. The *anntoationProperties* value should specify all the properties required by that annotation type in a delimited string. These are listed in the "Annotations File Format" section of this manual (see page 65).

The convenience method, getDelimiter() can be called to identify the delimiter that should be used.

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **modifyAnnotation(*label*, *annotationProperties*)**

```
ViewONE.modifyAnnotation(
    "myHighlight1",
    "X=500<P>Y=350<P>WIDTH=400<P>HEIGHT=600<P>");
```

Modifies the properties of the annotation that has a label property matching the one specified. The *annotationProperties* value only needs to specify the properties that are to be changed or added. Properties for the various annotation types are listed in the "Annotations File Format" section of this manual (see page 65).

The convenience method, getDelimiter() can be called to identify the delimiter that should be used.

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **deleteAnnotation(*label*)**

ViewONE.deleteAnnotation("myHighlight1");

Deletes the annotation that has a label property matching the one specified.

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **deleteAllAnnotations(*type*, *page*)**

ViewONE.deleteAllAnnotations("highlight", 1);

Deletes the annotation of the specified type from the specified page. The *type* and *page* parameter values have the same rules as in the getNumAnnotations(*type*, *page*) method (see page 56).

Method: **getDelimiter()**

var strDelimiter = ViewONE.getDelimiter();

Returns the string ViewONE is using for its delimiter when it returns multiple values in a string.

Method: **setDelimiter(*delimiter*)**

ViewONE.setDelimiter("||");

Sets the string ViewONE should use for its delimiter when it returns multiple values in a string.

Method: **parseProperty(*property*, *annotationProperties*)**

```
var strWidth = ViewONE.parseProperty(
        "width",
```

”[HIGHLIGHT]<P>PAGE=0<P>X=300<P>Y=350<P>WIDTH=400<P>HEIGHT=500<P>
LABEL=myHighlight1<P>HYPERLINK=<page><3><P>”);

Returns the value of a specified property from an annotation property list. This method should be used in combination with the getAnnotation(*label*) method and is really just a convenience method to save parsing in JavaScript.

Method:

startAnnotation(*type*)

ViewONE.startAnnotation(“text”);

Gets ViewONE to behave as if one of its annotations toolbar buttons has been selected. Typically this means that the mouse cursor is put into the mode where it is ready to place an annotation. The type of annotation is determined by the *type* parameter. The accepted *type* values are…

line
arrow
text
textSolid
note
highlight
highlightPoly
hyperlink
rectangle
square ***(V3-Only)***
redact
redactPoly
poly
openPoly
oval
circle ***(V3-Only)***
freehand
ruler ***(V3-Only)***
angle ***(V3-Only)***
anglereversed ***(V3-Only)***
stamp
stampMenu<*N*>

Where “stampMenu<*N*>” is used, *N* represents the position of the stamp item in the stamp menu list. The highest item in the menu is “1”, the next is “2” and so on.

V3 security note: This method is disabled by default unless the “annotationJavascriptExtensions” parameter is set to “true”.

Method:

startAnnotationWithProperties(*type, properties*)

(V3-Only)

ViewONE.startAnnotationWithProperties("text", "<color=blue>");

This method behaves the same as startAnnotation() but it also allows specific annotations properties to be defined (which override the defaults).

The format of the properties string is as follows:

"<property1=value1><property2=value2>…

e.g "<color=blue>", or "<color=blue><rotation=45>"

The optional properties are as follows…

Property name	Description and value options
Color	Color of text or line (or transparency color for image stamps)
FillColor	Background Color of text or fill color of shape
Menu	Menu text to display (for stamp menus)
Rotation	Rotation of annotation (0, 45, 90, 180 or 270)
Label	Text for the annotation label
HelpText	Tooltip text
Owner	Owner (username) text
FontHeight	Height in pixels for text/stamp annotations
Print	True = other users can print annotation False = other users cannot print annotation
Read	True = other users can read annotation False = other users cannot read annotation
Modify	True = other users can modify annotation False = other users cannot modify annotation
Delete	True = other users can delete annotation False = other users cannot delete annotation
ModifySecurity	(for annotation security model 2 – see page 41) True = other users can modify annotation security False = other users cannot modify annotation security

V3 security note: This method is disabled by default unless the "annotationJavascriptExtensions" parameter is set to "true".

Method: **setStickyAnnotations(*boolean*)**

ViewONE.setStickyAnnotations(true);

When this method is called with “true”, the user may re-use the selected annotation without re-selecting from the annotation toolbar.

When the method is called with “false”, the user must re-select the annotation to re-use. This method must be called before the user selects an annotation and only needs to be called once in a session.

Method: **addAnnotationStamp(*url, displayText*)**

ViewONE.addAnnotationStamp(“http://mysite/mystamp.tif#2, “Sign John Smith”);

Calling this method adds a new stamp to the annotation stamp menu. If the method has been called previously, the previous menu item is replaced rather than a new one being created.

The URL parameter specifies a String containing the URL of the image to be used as the annotation stamp, it can point to any image file that can be decoded by ViewONE. For a full list of the available file types supported by ViewONE see the user manual.

The # modifier can be used to specify a page to be extracted from the image file, for example mystamp.tif#2 will extract page 2 of mystamp.tif for use as the signature. If no # modifier is specified for a multi-page image then the first page will be used.

The displayText parameter specifies the String value that will be displayed in the annotation stamp menu for the stamp being added.

Annotation Security

This section of the manual describes the two methods available for implementing annotation security. Implementers are free to choose whichever method is best suited to their security requirements.

Simple Annotation Security

Simple annotation security uses the “edit” (see page 66) and “view” (see page 68) annotation properties to define whether or not a user can edit, view or print a particular annotation.

The “edit” annotation property is not included, by default. Therefore, when a user creates an annotation (using the user-interface), then saves, the 'annotations definitions file' will contain details about the annotation which will not include the "edit" property because that user has 'edit' privileges for that annotation. Then when another user retrieves the annotation file, again, without the "edit" property being specifically included, that user can edit that annotation.

However, the server object that serves up the annotation file may insert the 'edit' property for any annotation, and by setting it to 0 (zero) will prevent that user from editing that particular annotation. This can be done selectively for each annotation.

To make use of this functionality the assumption is made that you have some kind of server object (an exe, asp, JSP etc) that allows you to control the content of the annotation file when it is served out to users.

There is also a HTML parameter option "annotateEdit", described on page 17 that can be used to disable the edit option for all annotations. This is a global option and when used, the user concerned will not be able to edit/modify any annotation.

The same principle can be applied for the “view” annotation property. The default value is “3”, allowing all users to both view and print the annotation. Annotations created by the user will have this default value and when saved the “view” property will not be set in the annotations definition file.

The server object can then set the view property to “1”, to allow viewing of the annotation but not printing or “2” to allow printing but not viewing.

Extended Annotation Security

The simple annotation security options (see above) are somewhat limited in that users do not have any control over what privileges are assigned to annotations they create. Also, there is no way to individually control modification, deletion and reading privileges separately.

It is possible to enable 'extended' options that provide both additional user options and the ability to specify more detailed privileges.

The extended option relies on you using the "userId" HTML parameter (see page 25), then adding a HTML parameter called "annotationSecurityModel" (see page 41) with a value of two.

Example HTML:

<param name="userId" value="user1”>
<param name="annotationSecurityModel" value="2">

This will enable, as default, extended security options for each new annotation created by the user.

When the user then saves annotations, extra properties will be written to the annotations file, as follows...

SECURITYMODEL = 2
READ = 1
MODIFY = 1
EXECUTE = 1
PRINT = 1
DELETE = 1
*PASSWORDMODIFY = ******
PASSWORDSECURITY = *****
OWNER = Whoever*
MODIFYSECURITY = 1

You will note that there are separate privileges for read, modify, execute, print, delete and modify-security. Each of these can be changed by either the user or the server object (when serving the file back to the user).

By default, they are all set to 1 (enabled). The "MODIFYSECURITY" property indicates whether the user has the ability to edit these privileges using the user interface. If enabled, when the user selects the annotation, an additional 'padlock' context button will appear. If they click on this button and they will see a 'security' dialog permitting them to change any of the security properties above.

The owner of an annotation is the user that created the annotation in the first place, identified by the "owner" property also written to the annotations file. If this owner property matches the "userId" HTML parameter, then that user always has access to this security dialog.

Alternatively, the HTML parameter "userAdmin" (see page 42) can be specified. If set to "true", then irrespective of the "owner" and "userId" values, that user will have 'admin' privileges, and so be able to edit annotation security for any annotation.

So, if a user creates an annotation, then pulls up this dialog and un-ticks the 'modify' option, no user (except themselves and admin users) will be able to modify that annotation (e.g. they can't select it, move it etc). If modify was left ticked, but the 'delete' option was un-ticked, then other users can modify the annotation but they can't delete it. If the 'read' option is disabled, the other users will not get to see the annotation. If the 'print' option is disabled then users will not be able to print the annotation.

The 'execute' option is used for annotation 'hyperlinks'. i.e. if an annotation has a hyperlink, but 'execute' is disabled, then other users will not be able to use that hyperlink.

If a document has existing annotations that were created using the 'simple' method, then the extended options can be enabled on them by adding the "SECURITYMODEL = 2" line to each annotation (at retrieval time) by the server object serving the annotations file.

If the “annotationEnablePW” parameter is set to “true” (see page 42) then it is possible to set a password that must be entered before a user modify an annotaton and/or modify annotation security.

The passwords are saved in the annotations file, using the properties PASSWORDMODIFY and PASSWORDSECURITY, They are private-key 32-bit encrypted so they will not be able to viewable or editable except through ViewONE. If a user forgets a password assigned to an annotation then if they are the OWNER or an ADMIN user than they can clear and re-enter a password. All other users cannot, unless they enter the correct modify annotation security password.

The passwords are not displayed anywhere (the "*" character is used during entry and on the new dialog).

Annotations File Format

This section of the manual describes the structure of the text file that is used to define annotations. This is the file that must be supplied to ViewONE for viewing annotations and is the file format sent to the server by ViewONE.

General structure

The annotations file is an ASCII text file with the optional extension of "ANT". Annotations are defined through a set of properties, some mandatory, some optional, each one defined on a separate line.

In general the file is case insensitive except for where text properties are defined. Blank lines are ignored.

The format of this file is similar in nature to a Windows INI file.

The start of an annotation definition is denoted by a left square bracket. This is then followed by a "type" identifier (which indicates what kind of annotation is being defined – see below). The identifier is terminated by a right square bracket. A carriage-return, a linefeed, or a carriage-return and linefeed then delimit the end of the line.

Properties specific to the type of annotation defined then follow. Only one property can be defined per line.

For example…

```
[ARROW]
X1 = 535
Y1 = 277
X2 = 89
Y2 = 138
PAGE = 1
EDIT = 1
COLOR = 255, 0, 0
```

The end of an annotation definition is denoted by the start of another annotation definition (identified by the [and] brackets) or by the end of the file.

If no annotations are present in the document when annotations are saved, ViewONE returns a file with just [EMPTY] inside. When an annotations definition file is loaded it must contain either annotation definitions or the [EMPTY] marker. If it does not then an error message will be displayed and the document will not be shown.

Standard mandatory properties

Standard mandatory properties are now described. These properties are required for all annotation types.

page

PAGE = *n*

Specifies a page for the annotation.

If the value specified is –1 then the annotation is displayed across all pages in the document. To achieve this ViewONE produces a copy of the annotation for each page of the document. If annotations are then saved, these copies will be returned as individual annotations in the definitions file (along with their appropriate page number).

The LABEL properties (see "Standard Optional Parameters" – page 67) for these new annotations will be a copy of the original LABEL property with a page number appended. This makes these annotations unsuitable for use with hyperlinks, as their names will change automatically upon creation.

edit

EDIT = *1*

Boolean for whether the annotation can be edited or not. False=0, True=1.

Standard optional properties

Standard optional properties are now described. These properties are optional for any of the annotation types described above except where specifically excluded.

lineWidth

LINEWIDTH = *n*

Specifies the width of the annotation's lines in image pixels.

color

COLOR = *color*

The annotation's color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. The default values are 255, 0, 0 (red), but the default color can been changed using the "annotationsDefaultLineColor" or "annotationsDefulatTextColor" parameters.

label

LABEL = *any text*

Specifies an ID for the annotation. Carriage-returns and linefeeds are not permitted within the value. The default value is the annotation type; for example, "Arrow", "Oval", etc. Leading and trailing white space is removed upon save and load operations.

tooltip

TOOLTIP = *any text*

Specifies a tooltip for the annotation. Carriage-returns and linefeeds are not permitted within the value. The value is case sensitive. There is no default value.

Note: tooltips are particularly useful when used with annotation hyperlinks (see hyperlink section – page 71).

createDate

CREATEDATE = *dd mmm yyyy*, *hh*:*mm*:*ss*[, *loc*±*hh*:*mm*]

For example…

CREATEDATE = 12 Jun 2001, 18:27:05
CREATEDATE = 12 Jun 2001, 18:27:05, GMT+01:00

Specifies the date the annotation was created. The optional ", *loc*±*hh*:*mm*" is a local time zone stamp. "*loc*" stands for location, and is a three letter identifier for the time zone. The "*hh*:*mm*" specifies the amount the local time was offset from that location's time zone. The "+" or "-" then specify whether that time is ahead or behind the time zone.

modifiedDate

MODIFIEDDATE = *dd mmm yyyy*, *hh*:*mm*:*ss*[, *loc*±*hh*:*mm*]

For example…

MODIFIEDDATE = 12 Jun 2001, 18:27:05
MODIFIEDDATE = 12 Jun 2001, 18:27:05, GMT+01:00

Specifies the date the annotation was last modified. The optional ", *loc*±*hh*:*mm*" is a local time zone stamp. "*loc*" stands for location, and is a three letter identifier for the time zone. The "*hh*:*mm*" specifies the amount the local time was offset from that location's time zone. The "+" or "-" then specify whether that time is ahead or behind the time zone.

view

VIEW = *n*

Specifies an integer value for the print and view properties of an annotation. The possible values are as follows…

1 = viewing only (no printing)
2 = printing only (no viewing or editing by the user)
3 = printing and viewing

The default value is 3.

ViewONE does not return the property in the definition file if the value is 3.

pageSize

PAGESIZE = *n*, *n*

Specifies the main page's width and height associated with the annotation. The value breaks down into width and height in that order (from left to right).

It is only required if annotations are to be displayed in thumbnails where thumbnails are specified as separate files (using the "thumb<*N*>" HTML parameter – see ViewONE's HTML manual). In such cases, the width and height values must be those for the original main image (not the scaled thumbnail image). ViewONE is then able to rescale any annotations that may be present on that page to suite the thumbnail image.

pageURL

PAGEURL = *pageURL*

Shows the URL ViewONE used to retrieve the page the annotation was placed on.

The property is written out by ViewONE, but is ignored upon reading. It allows server-side processes that handle the annotations definition file to see which image file each annotation (and each page) is associated with.

createdId

CREATEDID = *userID*

Specifies the userID value that was in use when the annotation was created. The "userID" parameter is described in the "Applet Parameters: User options" section of this manual (see page 25).

If the userID value is not set at the time the annotation is saved the property is not returned in the definitions file.

modifiedId

MODIFIEDID = *userID*

Specifies the userID value that was in use when the annotation was last modified. The "userID" parameter is described in the "Applet Parameters: User options" section of this manual (see page 25).

If the userID value is not set at the time the annotation is saved the property is not returned in the definitions file.

blankOutImage

BLANKOUTIMAGE = *1*

Boolean for whether the page the annotation is on should be blanked out, rather than displayed. It is a security option, allowing an image to be blocked from view, but for its annotations to still be visible.

For example, if a user cannot view a particular page of a document then a simple text annotation could be setup on that page which says "Sorry, viewing denied" and has a BLANKOUTIMAGE value of 1.

The user will not be able to edit or modify this annotation, even if the EDIT property is set to 1. Do not include this property unless the value is 1.

customProperty

CUSTOMPROPERTY = *any text*

Specifies some custom text to be kept with the annotation. It is not interpreted by ViewONE, but if it is defined, as ViewONE reads the annotations file in, it makes a note of the entry and writes it out again at save time.

Just as the CUSTOM annotation type allows information to be associated with an annotation definition file, the CUSTOMPROPERTY allows information to be associated with individual annotations.

Note: the property is only written back at save time if it came from a valid annotation definition.

hyperlink

HYPERLINK = <*annotation type*><*hyperlink property*>[<*hyperlink property*>]

ViewONE (version 2.0) introduces the concept of “annotation hyperlinks”.

An annotation hyperlink is an action that is taken when an annotation is “clicked“ on by the user (or double-clicked – if the “annotationDblClick“ HTML parameter is used, or if ViewONE has the annotations toolbar open).

This action can be any of four types: to change to a specific page, to go to a specific annotation, to call a JavaScript function or to open a web page. Each of these types has additional parameters, described in the following pages.

We have tried to make setting up and using annotation hyperlinks as simple as possible (the easiest way to set them up is to use ViewONE’s user interface - see context sensitive toolbar for annotations). The following pages describe the format of the hyperlink properties.

Before trying to define your own hyperlinks in a definitions file it is worth having a play with the ViewONE user interface to see how they work. You may find that, for example, a JavaScript hyperlink is more suitable for some tasks than the web hyperlink. Both can be used to open a web page (for example), but using a JavaScript hyperlink would mean more flexibility over what that window looks like.

JavaScript hyperlinks can also be used for any other task, such as opening another document, or some custom job. JavaScript hyperlinks however, require the use of ViewONE’s JavaScript event-handler (see ViewONE’s JavaScript manual) and so are a little more complex to initially setup.

Using ViewONE’s annotation tooltips to help with hyperlinks

Any annotation can have a tooltip (see “Standard Optional Properties” section – page 67).

A tooltip is some text that appears when the user “hovers” the mouse over an item (in this case an annotation). If you have a hyperlink defined for that annotation, for example, to jump to another page in the document, then it would be helpful to the user if a tooltip was also defined for that annotation. Perhaps to inform the user of what will happen when the annotation is clicked upon. For example, “Click here to see page 5”, or “Click here to see the index”, and so on.

The four types of hyperlink are as follows…

Page hyperlinks	Hyperlinks between pages in a document (internal document hyperlinks).
Annotation hyperlinks	Hyperlinks between annotations in a document (internal document hyperlinks).
JavaScript hyperlinks	Hyperlink to call the JavaScript event handler.
Web page hyperlinks	Hyperlink to another web page (external hyperlink).

Page hyperlinks

HYPERLINK = <page><17>

Page hyperlinks require the text <page> followed by the <*page number*>. Note that with all hyperlink properties the "<" and ">" must be included as shown. Pages outside of the document's page range are ignored.

Annotation hyperlinks

HYPERLINK = <annotation><highlight1>

Annotation hyperlinks require the text <annotation> followed by the <*annotation label*> of the other annotation that it is to be linked to.

If a link is made to a label name shared by two or more annotations within a document, ViewONE will link to the first of those annotations.

When this hyperlink is used, ViewONE will scroll the page to the top-left of the linked annotation (or as close as it can get), while using the current scale preference (fit to height, fit to width, etc.).

However, if the linked annotation has any window settings associated with it, then they will be used instead. For example, a particular zoom factor, scroll position, rotation angle, etc. See description of the HYPERLINKSETTINGS property.

JavaScript hyperlinks

HYPERLINK = <javascript><help text>

JavaScript hyperlinks require the text <javascript> followed by <*any text*>. When the annotation is clicked ViewONE calls the designated JavaScript event handler (see JavaScript manual on how to define a JavaScript event handler). The <*any text*> parameter is used as the event handler's "text" property, and can be interpreted in any way you choose (e.g., as a document id, web page, comment, etc.). The event id is 23.

This hyperlink is useful for designing in an implementation specific hyperlink behavior. For example, upon receiving the hyperlink's text string, the JavaScript event handler could display a pop-up with help information, tell the server something, or even use ViewONE JavaScript calls to open a new document (see ViewONE's JavaScript manual).

Web page hyperlinks

HYPERLINK = <web><http://www.mysite.com/page.html><_self>

Web page hyperlinks require the text <web> followed by the <*web page url*> then the <*target*> for the new web page. The target can be any of the four default values assigned by ViewONE (see Appendix C) or a custom target, as defined by the "annotationTarget" HTML parameter (see page 31).

The target allows the choice of which "window" will display the new web page; <_self> for example, will cause the browser to change the current web page (containing ViewONE), where as <_blank> will open a new window containing the new web page.

The web page URL can be relative to the ViewONE's codebase parameter or it can be absolute. If the HTML parameter "annotationHyperlinkWeb" is used, then when a relative URL is used it will be relative to the base address as defined by that HTML parameter (see page 27).

hyperlinkSettings

HYPERLINKSETTINGS = <*settingsValue*>

This property defines “window” settings for an annotation…

If this property is included with an annotation and that annotation is linked to by another (by using the hyperlink property of the other annotation to link to this annotation), then the window settings defined by this property will be used on activation of that hyperlink.

The format of this property is quite lengthy, but it allows a wide range of window settings to be defined. If you are unsure what value to use for this property, the easiest way to get a value is to use the ViewONE user interface.

Select an annotation that you want to link to. Zoom in, scroll, rotate, etc. so that the page appears in the way you wish it to be viewed.

Then use the right mouse button to click on the annotation (to select it in edit mode), and click on the “H” hyperlink button of the context toolbar that appears.

Click on the “Grab window settings” button, save the annotation. You need to note the annotations “label” so that you know which annotation to link to when you define the hyperlink from another annotation.

Now look at the annotations definition file that ViewONE has created (this could be on the server if you use the “annotationSave” HTML parameter, or in the “ANT” file if the image was loaded from a local file).

Locate the annotation then you’ll find the HYPERLINKS value included for the window settings you used above.

The format for this property is as follows…

*view#scale#*flip*#rotation#invert#zoom#scrollx#scrolly#brightness#contrast#luminance*

This is where…

view	currently ignored but reflects the view mode used (use a value of 1)
scale	0 (fit to width) or 1 (fit to height) or 2 (best fit) or 3 (zoom)
flip	0 (no flip) or 1 = flip horizontally or 2 = flip vertically or 3 (flip both ways)
rotation	0 or 90 or 180 or 270 (angle in degrees)
invert	0 (not color inverted) or 1 (colors inverted)
zoom	zoom factor (e.g. 2.0)
scrollX	percentage of image to scroll in the x-axis
scrollY	percentage of image to scroll in the y-axis
brightness	0 to 512 (where 0 =dark, 512 = light and 256 is the middle typical value)
contrast	0 to 512 (256 is the middle typical value)
luminance	0 to 512 (256 is the middle typical value)

Example value: `1#3#0#90#1#2.0#20#14#256#256#512`

Annotation types and their properties

Annotation specific properties are now described. For each annotation type there are both mandatory and optional properties.

The standard optional properties described in the previous section apply to all annotation types except where specifically excluded (for example, LINECOLOR).

Some default values for both specific and standard properties can be changed using HTML parameters described in the "Applet Parameters: Default options" section of this manual (see page 25).

Description:

Arrow

Defines an arrow annotation. There are two points in the definition. The arrow head is drawn at the first point.

Definition:

[ARROW]

(Mandatory)

X1 = *n*	X coordinate of the arrow's first point. *n* is an image pixel value on the image's X axis.
Y1 = *n*	Y coordinate of the arrow's first point. *n* is an image pixel value on the image's Y axis.
X2 = *n*	X coordinate of the arrow's second point.
Y2 = *n*	Y coordinate of the arrow's second point.

(Optional)

ARROWHEADSIZE = *n*	Specifies the arrow's head size. *n* has a minimum value of 1 and no fixed upper value. The default value is 1.

See standard optional properties…

The minimum LINEWIDTH for this annotation is 1.

Example:

```
[ARROW]
X1 = 537
Y1 = 277
X2 = 89
Y2 = 138
PAGE = 1
EDIT = 1
ARROWHEADSIZE = 1
```

Description:

Custom

Defines a custom annotation type. The custom annotation type is not really an annotation, but should be thought of more as way to add extra information to an annotations definition file.

This annotations type cannot be added from the user interface, but can be generated through a server-side process that adds instances to a definitions file. An instance is not interpreted by ViewONE, but if one is defined, as ViewONE reads the annotations file in, it makes a note of the entry and writes it out again at save time (though it is not certain the entry will be in the same place when written out again).

The CUSTOM type allows server-side processes to keep track of custom information between definition time and save time, without ViewONE interpreting it in the meantime.

Definition:

[CUSTOM]

(Mandatory)

There are no mandatory properties.

(Optional)

There are no optional properties that apply.

Example:

```
[CUSTOM]
This text shows that you can put anything into a CUSTOM annotation.
Text 1
Text 2
Text 3
I was generated by server "nautilus1" @ 18:31 GMT on 27/03/02
```

Description:

Freehand

Defines a freehand line. There must be a minimum of three points in the definition.

Definition:

[FREEHAND]

(Mandatory)

X1 = *n*	X coordinate of the line's first point. *n* is an image pixel value on the image's X axis.
Y1 = *n*	Y coordinate of the line's first point. *n* is an image pixel value on the image's Y axis.
X2 = *n*	X coordinate of the line's second point.
Y2 = *n*	Y coordinate of the line's second point.
X3 = *n*	X coordinate of the line's third point.
Y3 = *n*	Y coordinate of the line's third point.

(Optional)

Xn = *n*	X coordinate of point *n* on the line.
Yn = *n*	Y coordinate of point *n* on the line.

See standard optional properties…

The minimum LINEWIDTH for this annotation is 1.

Example:

```
[FREEHAND]
X1 = 408
Y1 = 940
X2 = 408
Y2 = 940
X3 = 412
Y3 = 936
X4 = 412
Y4 = 927
X5 = 425
Y5 = 907
PAGE = 1
EDIT = 1
```

Description: **Highlight Rectangle**

Defines a rectangular highlight.

Definition: [HIGHLIGHT]

(Mandatory)

X = *n*	X coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's X axis.
Y = *n*	Y coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's Y axis.
WIDTH = *n*	Width of the rectangle in image pixels.
HEIGHT = *n*	Height of the rectangle in image pixels.

(Optional)

LINEWIDTH = 0	The LINEWIDTH property will be ignored if it is not set to 0.
FILLCOLOR = *color*	The highlight color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. If no color is defined then the highlight is invisible.
TRANSPARENT = *0*	Boolean for whether the highlight is transparent. False=0, True=1. The default value is 0.
ASPECTRATIO=1:1	When set to 1:1 will force the highlight to be a square

See standard optional properties…

The minimum LINEWIDTH for this annotation is 0 (default).

Example:

```
[HIGHLIGHT]
X = 674
Y = 61
WIDTH = 310
HEIGHT = 176
PAGE = 1
EDIT = 1
LINEWIDTH = 0
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
```

Description: **Highlight Polygon**

Defines a highlight polygon. There must be a minimum of three points in the definition.

Definition: [HIGHLIGHTPOLYGON]

(Mandatory)

X1 = *n*	X coordinate of the polygon's first point. *n* is an image pixel value on the image's X axis.
Y1 = *n*	Y coordinate of the polygon's first point. *n* is an image pixel value on the image's Y axis.
X2 = *n*	X coordinate of the polygon's second point.
Y2 = *n*	Y coordinate of the polygon's second point.
X3 = *n*	X coordinate of the polygon's third point.
Y3 = *n*	Y coordinate of the polygon's third point.

(Optional)

Xn = *n*	X coordinate of point n in the polygon.
Yn = *n*	Y coordinate of point n in the polygon.
LINEWIDTH = 1	The LINEWIDTH property will be ignored if it is not set to 1.
FILLCOLOR = *color*	The highlight color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. If no color is defined then the highlight is invisible.
TRANSPARENT = *0*	Boolean for whether the highlight is transparent. False=0, True=1. The default value is 0.

See standard optional properties…

The minimum LINEWIDTH for this annotation is 0 (default 1).

Example:

```
[HIGHLIGHTPOLYGON]
X1 = 1246
Y1 = 61
X2 = 1581
Y2 = 220
X3 = 1356
Y3 = 371
X4 = 1144
Y4 = 200
PAGE = 1
EDIT = 1
LINEWIDTH = 1
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
```

Description: **Line**

Defines a line. There are two points in the definition.

Definition: [LINE]

(Mandatory)

X1 = *n*	X coordinate of the line's first point. *n* is an image pixel value on the image's X axis.
Y1 = *n*	Y coordinate of the line's first point. *n* is an image pixel value on the image's Y axis.
X2 = *n*	X coordinate of the line's second point.
Y2 = *n*	Y coordinate of the line's second point.

(Optional) See standard optional properties…

The minimum LINEWIDTH for this annotation is 1.

Example:

```
[LINE]
X1 = 498
Y1 = 167
X2 = 85
Y2 = 65
PAGE = 1
EDIT = 1
```

Description: **Note**

Defines a note and its text.

Definition: [NOTE]

(Mandatory)

X = *n*	X coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's X axis.
Y = *n*	Y coordinate of the polygon's first point. *n* is an image pixel value on the image's Y axis.
WIDTH = *n*	Width of the rectangle in image pixels.
HEIGHT = *n*	Height of the rectangle in image pixels.
TEXT = *any text*	Specifies text for the note. New lines are indicated by the <N> identifier. Leading and trailing white space is removed upon save and load operations.

(Optional)

FILLCOLOR = *color*	The note's color (background color) specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. The default values are 255, 255, 153 (pale yellow).
TRANSPARENT = *0*	Boolean for whether the note is transparent. False=0, True=1. The default value is 0.
RECTANGULAR = *0*	Boolean for whether the note style is rectangular (i.e., without leaf turn up). The default is 0.

See standard optional properties… LINEWIDTH and HYPERLINK not available.

Example:

```
[NOTE]
X = 81
Y = 323
WIDTH = 50
HEIGHT = 65
TEXT = This is line 1<N>This is line 2
PAGE = 1
EDIT = 1
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
RECTANGULAR = 0
```

Description:

Open Polygon

Defines an open-ended polygon. There must be a minimum of three points in the definition.

Definition:

[OPENPOLYGON]

(Mandatory)

X1 = *n*	X coordinate of the polygon's first point. *n* is an image pixel value on the image's X axis.
Y1 = *n*	Y coordinate of the polygon's first point. *n* is an image pixel value on the image's Y axis.
X2 = *n*	X coordinate of the polygon's second point.
Y2 = *n*	Y coordinate of the polygon's second point.
X3 = *n*	X coordinate of the polygon's third point.
Y3 = *n*	Y coordinate of the polygon's third point.

(Optional)

Xn = *n*	X coordinate of point n in the polygon.
Yn = *n*	Y coordinate of point n in the polygon.

See standard optional properties…

The minimum LINEWIDTH for this annotation is 1.

Example:

```
[OPENPOLYGON]
X1 = 1242
Y1 = 670
X2 = 1246
Y2 = 878
X3 = 927
Y3 = 882
X4 = 1017
Y4 = 658
PAGE = 1
EDIT = 1
```

Description: **Oval**

Defines an oval.

Definition: [OVAL]

(Mandatory)

X = *n*	X coordinate of the oval's center. *n* is an image pixel value on the image's X axis.
Y = *n*	Y coordinate of the oval's center. *n* is an image pixel value on the image's Y axis.
WIDTH = *n*	The oval's horizontal radius in image pixels.
HEIGHT = *n*	The oval's vertical radius in image pixels.

(Optional)

FILLCOLOR = *color*	The oval's fill color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. The default values are 255, 255, 255 (white).
TRANSPARENT = *0*	Boolean for whether the oval is transparent. False=0, True=1. The default value is 1.
ASPECTRATIO=1:1	When set to 1:1 will force the highlight to be a circle **(V3-Only)**

See standard optional properties…

The minimum LINEWIDTH for this annotation is 0.

Example:

```
[OVAL]
X = 257
Y = 752
WIDTH = 143
HEIGHT = 130
PAGE = 1
EDIT = 1
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
```

Description: **Polygon**

Defines a closed polygon. There must be a minimum of three points in the definition.

Definition: [POLYGON]

(Mandatory)

X1 = *n*	X coordinate of the polygon's first point. *n* is an image pixel value on the image's X axis.
Y1 = *n*	Y coordinate of the polygon's first point. *n* is an image pixel value on the image's Y axis.
X2 = *n*	X coordinate of the polygon's second point.
Y2 = *n*	Y coordinate of the polygon's second point.
X3 = *n*	X coordinate of the polygon's third point.
Y3 = *n*	Y coordinate of the polygon's third point.

(Optional)

Xn = *n*	X coordinate of point n in the polygon.
Yn = *n*	Y coordinate of point n in the polygon.
FILLCOLOR = *color*	The polygon's fill color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. The default values are 255, 255, 255 (white).
TRANSPARENT = *0*	Boolean for whether the polygon is transparent. False=0, True=1. The default value is 1.

See standard optional properties…

The minimum LINEWIDTH for this annotation is 0.

Example:

```
[POLYGON]
X1 = 1528
Y1 = 457
X2 = 1634
Y2 = 694
X3 = 1442
Y3 = 809
X4 = 1360
Y4 = 621
PAGE = 1
EDIT = 1
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
```

Description: **Rectangle**

Defines a rectangle.

Definition: [RECTANGLE]

(Mandatory)

X = *n*	X coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's X axis.
Y = *n*	Y coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's Y axis.
WIDTH = *n*	Width of the rectangle in image pixels.
HEIGHT = *n*	Height of the rectangle in image pixels.

(Optional)

FILLCOLOR = *color*	The rectangle's fill color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. The default values are 255, 255, 255 (white).
TRANSPARENT = *0*	Boolean for whether the rectangle is transparent. False=0, True=1. The default value is 1.
ASPECTRATIO=1:1	When set to 1:1 will force the highlight to be a square **(V3-only)**

See standard optional properties…

The minimum LINEWIDTH for this annotation is 0.

Example:

```
[RECTANGLE]
X = 690
Y = 269
WIDTH = 282
HEIGHT = 209
PAGE = 1
EDIT = 1
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
```

Description: **Redaction**

Defines a rectangular redaction (a filled rectangle).

Definition: [REDACT]

(Mandatory)

X = *n*	X coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's X axis.
Y = *n*	Y coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's Y axis.
WIDTH = *n*	Width of the rectangle in image pixels.
HEIGHT = *n*	Height of the rectangle in image pixels.

(Optional)

LINEWIDTH = 0	The LINEWIDTH property will be ignored if it is not set to 0.
FILLCOLOR = *color*	The redaction's color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. The default values are 0, 0, 0 (black).
TRANSPARENT = *0*	Boolean for whether the redaction is transparent. False=0, True=1. The default value is 0.
ASPECTRATIO=1:1	When set to 1:1 will force the highlight to be a square

See standard optional properties…

The minimum LINEWIDTH for this annotation is 0 (default).

Example:

```
[REDACT]
X = 1058
Y = 371
WIDTH = 225
HEIGHT = 221
PAGE = 1
EDIT = 1
LINEWIDTH = 0
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
```

Description:

Redaction Polygon

Defines a redaction polygon (a filled polygon). There must be a minimum of three points in the definition.

Definition:

[REDACTPOLYGON]

(Mandatory)

X1 = *n*	X coordinate of the polygon's first point. *n* is an image pixel value on the image's X axis.
Y2 = *n*	Y coordinate of the polygon's first point. *n* is an image pixel value on the image's Y axis.
X2 = *n*	X coordinate of the polygon's second point.
Y2 = *n*	Y coordinate of the polygon's second point.
X3 = *n*	X coordinate of the polygon's third point.
Y3 = *n*	Y coordinate of the polygon's third point.

(Optional)

Xn = *n*	X coordinate of point n in the polygon.
Yn = *n*	Y coordinate of point n in the polygon.
LINEWIDTH = 1	The LINEWIDTH property will be ignored if it is not set to 1.
FILLCOLOR = *color*	The redaction's color specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. The default values are 0, 0, 0 (black).
TRANSPARENT = *0*	Boolean for whether the redaction is transparent. False=0, True=1. The default value is 0.

See standard optional properties…

The minimum LINEWIDTH for this annotation is 0 (default 1).

Example:

```
[REDACTPOLYGON]
X1 = 702
Y1 = 519
X2 = 948
Y2 = 621
X3 = 845
Y3 = 792
X4 = 645
Y4 = 756
PAGE = 1
EDIT = 1
LINEWIDTH = 1
FILLCOLOR = 255, 255, 255
TRANSPARENT = 0
```

Description: **Stamp**

Defines an image overlay (an overlaid image which may be in any of the image formats supported by ViewONE).

Definition: [STAMP]

(Mandatory)

X = *n*	X coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's X axis.
Y = *n*	Y coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's Y axis.
RESOURCE = image:*image file*	Path (absolute or relative) to the stamp's image file. The path must have the "image:" prefix. Spaces are permitted in the path.

(Optional)

SCALE = *n*	Specifies the scale multiplier. 100% = 1.0, 200% = 2.0, and so on. The default value is 1.0.
RESOURCE = image:mytif.tif<*color*>	A comma delimited RGB value (or ViewONE color scheme text) can be added to the end of the RESOURCE property to specify the transparent color for the stamp. The values need to be contained within the < and > delimiters. This color will then be treated as transparent and the background image will be visible through the areas where this color occurs. This can be a convenient way to overlay transparent images, such as logos, watermarks, etc.
ROTATION = *n*	The stamp's rotation specified in degrees. Valid values for *n* are 0, 90, 180, 270, 360. The default is 0.
FONTHEIGHT = *n*	Specifies the font height to use for the stamp text. *n* is an image pixel value. The minimum acceptable value is 1.

See standard optional properties… COLOR and LINEWIDTH not available.

Example:

```
[STAMP]
X = 134
Y = 1017
SCALE = 1.0
RESOURCE = image:mytif.tif<255, 255, 255>
ROTATION = 180
PAGE = 1
EDIT = 1
LINEWIDTH = 0
```

Description: **Text**

Defines a text overlay.

Text annotations are displayed using a predefined font. There are several fonts, each based on Arial and of different sizes. It is important to note that ViewONE does not rely on the font configuration of the user's machine. The font is loaded from a font resource file specific to ViewONE that is supplied as part of the ViewONE installation.

Font size is specified through the FONTHEIGHT property.

Two properties are provided to allow for rotation of text annotations so that the most appropriate one can be used when integrating into existing systems. TEXTROTATION is the recommended property.

Definition: [TEXT]

(Mandatory)

X = *n*	X coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's X axis.
Y = *n*	Y coordinate of the rectangle's top left corner. *n* is an image pixel value on the image's Y axis.
TEXT= *any text*	Specifies text for the overlay. New lines are indicated by the <N> identifier. Leading and trailing white space is removed upon save and load operations.
FONTHEIGHT = *n*	Specifies the font height to use for the overlay's text. *n* is an image pixel value. The minimum acceptable value is 1.

(Optional)

FILLCOLOR = *color*	The overlay's fill color (background color) specified in either comma delimited RGB values or the ViewONE color scheme text, as listed in Appendix A. This property only comes into effect where the TRANSPARENT property is set to 0. The default values are 255, 255, 153 (pale yellow). If the TRANSPARENT property is set to 1 and the PAGE specified is a color image, the text will be given a visible fill color using the values 255, 255, 255 (white), as color transparencies are not supported.
ROTATION = *n*	The text's rotation specified in degrees. Valid values for *n* are 0, 90, 180, 270, 360. This differs from the TEXTROTATION property in that the rotation is clockwise relative to the page. The default is 0.
TEXTROTATION = *n*	The text's rotation specified in degrees. A valid value is an integer between 0 and 359. This differs from the ROTATION property in that the rotation is counterclockwise relative to the page. The default is 0.

TRANSPARENT = *0*	Boolean for whether the overlay is transparent. False=0, True=1. The default value is 1.
SEMITRANSPARENT = *0*	Boolean for whether the text is semi-transparent. False=0, True=1. The default value is 0. If the text is set to semi-transparent then the color used for the overlay's fill color will also be forced to semi-transparent.

See standard optional properties… LINEWIDTH not available.

Example:

[TEXT]
X = 85
Y = 265
TEXT = This is line 1<N>This is line 2
FONTHEIGHT = 34
PAGE = 1
EDIT = 1
FILLCOLOR = 255, 255, 0
ROTATION = 0
TEXTROTATION = 90
TRANSPARENT = 0
SEMITRANSPARENT = 0

Annotations Server-side 'Save' Object

This section of the manual describes the server-side object required to receive annotations sent by ViewONE.

General description

The object is called when an annotations save operation is carried out by ViewONE. Its purpose is to receive the annotation definitions that a user may have created or edited.

Sample objects are provided by Daeja (see Appendix E), however it is likely that whoever implements the system will want to modify them or write an object of their own. The object can be written in any server-side object style (CGI, ASP, EXE, servlet, etc), as it only needs the ability to receive and parse an annotations text stream.

With this approach there is the freedom to adapt the handling of definitions to meet implementation specific requirements. For example, the definitions could be written to a database, they could be stored as text files next to their respective images, or they could be passed on again to another object.

The approaches that can be used to implement the annotations server-side save object are discussed over the next few pages.

HTTP:POST Approach (recommended)

The location of the POST object is specified using the “annotationSavePost” parameter (described on page 12). The value of this parameter must be in the form of a URL (relative to the codebase or an absolute value).

Parameters may be added to the POST data, such as user ID or document ID to send user or document specific annotations, by using the “annotationPostPrefix” parameter. ViewONE will then tag on its own annotations data.

At save time the annotations data is URL encoded and put into a series of segments. Each segment can contain up to 64KB of encoded data. The number of segments generated depends on the amount of annotations data to be sent.

The following parameters are then added to the URL specified by the “annotationSavePost” parameter...

parameter	***description***
size	Size in bytes of the URL encoded annotations data being sent.
numdata	Number of data segments sent.
data<*N*>	URL encoded annotations data, where <*N*> is the sequential identifier for the segment.

The call to the server-side object is then made and the data passed with a content-type of “application/x-www-form-urlencoded”.

On the server-side, to retrieve the annotations data from the URL the data<N> segments are simply concatenated. URL encoded data is normally handled by the web server, so this should not be a concern. The data retrieved can then be checked by comparing the value of the size parameter passed in the URL against the size of the data actually received.

The HTTP:POST method is more efficient than the HTTP:GET method as a large quantity of annotations data can be passed to the server in a single call (in one segment 64 KB can be passed, whereas the HTTP:GET is limited to passing a maximum of 256 bytes per call).

It is also likely to be more reliable, as there will probably only be one call to the server involved. Whereas the HTTP:GET method will probably have to make several calls, highlighting any connection problems there might be.

Servlet Approach

The location of the servlet is specified using the "annotationSaveServlet" parameter (described on page 13). The value of this parameter must be in the form of a URL (relative to the codebase or an absolute value). Parameters may be added to the URL, such as user ID or document ID, in order to provide user or document specific annotations, but it is up to the object's author to process these.

A servlet server is required to run the annotations save servlet. There are a number of vendors with appropriate products. For example, the products from New Atlanta Communications at http://www.newatlanta.com.

We have prepared an example of an annotations server-side save servlet that uses this technique. It can be found in Appendix E and on our download pages at http://www.daeja.com/pub/start/downloads.html

This approach is recommended where HTTP:POST is not an option or where servlet servers are used.

HTTP:GET Approach

Where HTTP:POST and servlet usage is not an option then this approach may be best.

The location of the object is specified using the “annotationSave” parameter (described over the page). The value of this parameter must be in the form of a URL (relative to the codebase or an absolute value). Parameters may be added to the URL, such as user ID or document ID, in order to provide user or document specific annotations, but it is up to the object’s author to process these.

ViewONE will then tag on its own mandatory parameters to that URL (see below) so it is important to terminate the URL value correctly. For example, terminate your last parameter value with an ampersand (&), or if you have no parameters then terminate your script value with the question mark (?).

When a save operation is carried out the object may be called several times by ViewONE depending on the size of the definitions file. Each time the object is called it passes a portion of the annotations text file as a parameter in the URL. Thus, the file is effectively streamed to the object using only HTTP.

Since there is a limit of 256 characters on the length of the URL, the size of the portions sent will be smaller, as the object’s URL also uses some of this space. It is therefore advisable to keep the URL as short as possible.

Each time the object is called by ViewONE the object must send back a confirmation in the form of <OK> or <FAILED>. If the object reports <FAILED> ViewONE will retry several times. If the object continues to report a failure or if the connection is dropped then ViewONE will report this fact to the user.

While preparing the server-side object and to assist with addressing with any problems it is advisable to use the “annotationTrace” parameter described in a previous section. This parameter will allow monitoring of what ViewONE is sending to the object and to locate specific problems.

Parameters added by ViewONE

ViewONE will add the following parameters to the “annotationSave” URL…

seq=<*n*>&
of=<*n*>&
text=<*hex encoded ASCII string*>

seq	The number of the current text stream being sent to the server-side object.
of	The number of text streams making up the definitions file.
text	A portion of the definitions file converted to ASCII numerics represented as two-digit hexadecimal values (see below).

For example…

<PARAM NAME=“annotationSave”
value=“http://www.mysite.com/myobjects/save.pl?userID=12&docID=8791&”>

ViewONE will append to this as follows…

http://www.mysite.com/myobjects/save.pl?userID=12&docID=8791&seq=1&of=2&text=41424320

http://www.mysite.com/myobjects/save.pl?userID=12&docID=8791&seq=2&of=2&text=444546

The server-side object should append both text parameters to form a concatenated value, as follows…

41424320 + 444546 = 41424320444546

This represents the two-digit hexadecimal encoded ASCII string and can be decoded as follows…

Hex	41	42	43	20	44	45	46
Decimal	65	66	67	32	68	69	70
String	A	B	C	<SPACE>	D	E	F

Using the above table we can see that the encoded string translates to “ABC DEF”.

Why are text files encoded and sent in this way?

Text streams are encoded in this way to permit spaces and non-alphanumeric characters to be sent to the server-side object, using nothing but HTTP:GET methods. Such characters would not ordinarily be permitted within a URL, however by encoding them in this way we avoid this problem. By using HTTP:GET we also eliminate the need for specific networking software on the server, such as ASPs or servlets, thus making is more easily integrated into any environment.

There is a format (RFC1867), which has already been defined for uploading files via HTTP, however implementation of this would require specific server side components and networking software to be implemented. We have not ruled out this approach entirely and so may review whether implementation of this standard could be included in future versions. For the current release, a simple hexadecimal encoded ASCII has been implemented to ease the initial integration.

We have prepared a couple of examples of an annotations server-side save object that uses this technique, one written for CGI (Perl) and the other for ASP (VBScript). They can be found in Appendix E and on our support pages at http://www.daeja.com

Appendix A: ViewONE Color Scheme Text

Where parameters require a color value, it is possible to use either a comma delimited RGB value or the ViewONE color scheme text. The text values that can be used in place of RGB values within those parameters are listed below. The values are case insensitive.

Text	***Corresponding RGB values***
Black	0, 0, 0
Blue	0, 0, 255
Cyan	0, 255, 255
DarkGray	64, 64, 64
Gray	128, 128, 128
Green	0, 255, 0
LightGrey	192, 192, 192
Magenta	255, 0, 255
Orange	255, 200, 0
Pink	255, 175, 175
Red	255, 0, 0
White	255, 255, 255
Yellow	255, 255, 0

Appendix B: ViewONE Default Stamps

ViewONE's default stamp values and their properties are as follows…

annotationStamp parameter value	***annotationStampProperties parameter value***
<date>	<menu=Today's date>
RECEIVED	<menu=Received>
PAID	<menu=Paid>
APPROVED	<menu=Approved>
* REJECTED *	<menu=Rejected>
* VOID *	<menu=Void>

Appendix C: ViewONE Default Web Hyperlink Targets

ViewONE's default web hyperlink target values are as follows…

<target><menu label text>
<_self><Same window>
<_blank><New window>
<_parent><Parent frame>
<_top><Top frame>

Appendix D: Sample Annotations File

```
[HIGHLIGHT]
X = 247
Y = 158
WIDTH = 260
HEIGHT = 257
PAGE = 2
LINEWIDTH = 0
COLOR = 255, 255, 0
TRANSPARENT = 1
LABEL = highlight2
HYPERLINKSETTINGS = 1#3#0#0#0#1.953125#-27.511196417146515#-18.875119161105815#256#256#256
EDIT = 1
CREATEDATE = 15 Jun 2001, 20:27:12, GMT+01:00
MODIFIEDDATE = 15 Jun 2001, 18:42:38, GMT+01:00
PAGESIZE = 800, 537

[TEXT]
X = 915
Y = 73
FONTHEIGHT = 34
TEXT = This is a text annotation - line 1<N>and this is its second line
PAGE = 3
COLOR = 255, 0, 0
TRANSPARENT = 1
LABEL = text2
EDIT = 1
CREATEDATE = 15 Jun 2001, 18:42:59, GMT+01:00
MODIFIEDDATE = 15 Jun 2001, 18:43:28, GMT+01:00
PAGESIZE = 4871, 3325

[HIGHLIGHT]
X = 154
Y = 346
WIDTH = 405
HEIGHT = 251
PAGE = 1
LINEWIDTH = 0
FILLCOLOR = 255, 255, 0
COLOR = 255, 255, 0
TRANSPARENT = 1
LABEL = highlight1
TOOLTIP = Click here to see page 2 zoomed
HYPERLINK = <annotation><highlight2>
HYPERLINKSETTINGS = 1#1#0#0#0#0.3641176470588235#0.16155088852988692#0.0#256#256#256
EDIT = 1
CREATEDATE = 15 Jun 2001, 21:56:24, GMT+01:00
MODIFIEDDATE = 15 Jun 2001, 18:42:15, GMT+01:00
PAGESIZE = 1700, 2340

[ARROW]
X1 = 2426
Y1 = 619
X2 = 1957
Y2 = 251
PAGE = 3
LINEWIDTH = 50
COLOR = 255, 0, 0
LABEL = arrow1
EDIT = 1
CREATEDATE = 15 Jun 2001, 18:43:36, GMT+01:00
MODIFIEDDATE = 15 Jun 2001, 18:43:51, GMT+01:00
PAGESIZE = 4871, 3325
```

Appendix E: Sample Server-side 'Save' Objects

i) Java Servlet example

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.text.SimpleDateFormat;

// Sample servlet for receiving and saving annotations from ViewONE

public class annSave extends HttpServlet implements SingleThreadModel
{
   public void doGet( HttpServletRequest req, HttpServletResponse res ) throws IOException
   {
      res.setContentType( "text/html" );
      PrintWriter out = res.getWriter();
      out.println( "<CENTER><FONT SIZE=+1 FACE=\"arial\">ViewONE Servlet Just Ran Successfully
Again!</FONT></CENTER>" );
   }

   /**Process the HTTP Post request*/

   public void doPost(HttpServletRequest request, HttpServletResponse response)
                                                                        throws ServletException, IOException
   {
      //get parameters
      int fileLength = 0;
      int totalRead = 0;
      String readError = null;
      byte[] byteBuffer = null;

      String filename = request.getParameter("filename");

      try
      {
         //read data
         ServletInputStream inStream = request.getInputStream();
         fileLength = request.getContentLength();

         byteBuffer = new byte[fileLength];

         totalRead = inStream.read(byteBuffer);

         inStream.close();
      }
      catch (Exception e)
      {
         readError = e.toString();
         e.printStackTrace();
      }

      try
      {
         String responseText = "NOT OK";

         //check header matches data length
         if (fileLength > 0)
         {
```

```
                    //verify size
            if (fileLength != totalRead)
            {
               if (readError != null)
                  responseText = "NOT OK: Not enough data: " + totalRead + " of " + fileLength + "/" +
readError;
               else
                  responseText = "NOT OK: Not enough data: " + totalRead + " of " + fileLength;
            }
            else
            {
               if (filename == null)
               {
                   responseText = "NOT OK: No filename parameter";
               }
               else
               {
                  try
                  {
                     //save annotation in the filename we've been given
                     FileOutputStream fo = new FileOutputStream(filename);
                     fo.write(byteBuffer);
                     fo.flush();
                     fo.close();

                     responseText = "OK: " + totalRead + " bytes and saved in " + filename;
                  }
                  catch (Exception e)
                  {
                     responseText = "NOT OK: Unable to write to file " + filename + ": " + e;
                  }
               }
            }
         }
         else
         {
            responseText = "NOT OK: No Data!";
         }

         response.setContentType("text/plain");
         response.setContentLength(responseText.length());

         OutputStream out = response.getOutputStream();
         out.write(responseText.getBytes());
         out.close();
      }
      catch (Exception e)
      {
         e.printStackTrace();
      }
   }
}
```

ii) CGI/Perl example

```
#!/usr/bin/perl
################################################################################
#
# This is a simple perl cgi script to demonstrate reading a image file
# and sending it to ViewONE
#
# Note header details includes Content-Length. This is not mandatory
# but assists ViewONE in handling image retrieval and allows a percent
# bar to be displayed to the user
#
################################################################################

#$file = "test.log";
#Sample script, to receive and save annotations

#open a log file

$seq = 0;
$of = 0;
$originalname = "";
$text = "";
$success = "<OK>";
$one = "1";
$decodedtext = "";

$appendix = "";

&get_params;
&convert_text;

if ($seq eq $one)
{
        #first in sequence
        $openmmode = ">"; #create mode
}
else
{
        $openmmode = ">>"; #append mode
}

$originalname = $ENV{'REMOTE_HOST'};
$file = "$originalname$id.ant";

#last in the sequence
if (open(AFILE, $openmmode . $file))
{
        #append to file
        print AFILE $decodedtext;
        close AFILE;
}
else
{
        $success = "<FAILED>";
}

print "\n<$success>";
```

```
sub get_params
{
    # Split the name-value pairs
    @pairs = split(/&/, $ENV{'QUERY_STRING'});

    # For each name-value pair:
    foreach $pair (@pairs)
    {
        # Split the pair up into individual variables
        local($name, $value) = split(/=/, $pair);

        if ($name eq "seq")
        {
            $seq = $value;
        }

        if ($name eq "of")
        {
            $of = $value;
        }

        if ($name eq "name")
        {
            $originalname = $value;
        }

        if ($name eq "text")
        {
            $text = $value;
        }

        if ($name eq "id")
        {
            $id = $value;
        }
    }
}

sub convert_text
{
        # convert string of 2 character hex values into ascii chars

        $decodedtext = "";

        for ($charnum=0; $charnum<length($text);$charnum+=2)
        {
                $char = substr($text, $charnum, 2);

                if (hex($char) eq 10)
                {
                        # ignore this one
                }
                elsif (hex($char) eq 13)
                {
                        $decodedtext = $decodedtext . "\n";
                }
                else
                {
                        $decodedtext = $decodedtext . chr(hex($char));
                }
        }
}
```

iii) ASP/VBScript example

```
<%@ Language=VBScript%>
<% option explicit%>

<%
        '** Daeja Image Systems, sample ASP code for the "Annotation Save Object"
        '**
        '** The code is feely available for editing and distribution.
        '** It is not meant to be fully error proof but provides an example of
        '** how ASP can be used to receive annotation data sent by ViewONE, then
        '** save it locally.
        '**
        '** If you want to write annotation data to a database or other medium,
        '** see comments below that indicate where to edit the code.

        On Error Resume Next

        Response.Buffer = true
        Response.Expires = 0
        Response.Clear

        Const conIntForAppending        = 8
        Const conStrFirst               = "1"
        Const conStrExt                 = ".ant"
        Const conStrParamDelim          = "&"
        Const conStrValDelim            = "="
        Const conStrSeq                 = "seq"
        Const conStrOf                  = "of"
        Const conStrName                = "name"
        Const conStrText                = "text"
        Const conStrId                  = "id"

        Dim objFileSystem
        Dim objFile
        Dim strFilename

        Dim strSuccess
        Dim strSeq
        Dim strOf
        Dim strName
        Dim strText
        Dim strId
        Dim strDecodedText

        strFilename     = ""
        strSuccess      = "<OK>"
        strSeq          = ""
        strOf           = ""
        strName         = ""
        strText         = ""
        strId           = ""
        strDecodedText  = ""


        '** Retrieve parameters (sequence, name, text and id)

        Call getParams()

        '** Convert text from Hex-Encoded-ASCII back to straight ASCII

        Call convertText()

        '** Construct a local filename (this can be changed to anything suitable)
```

```
        strName = Request.ServerVariables ("REMOTE_ADDR")
        strFilename = strName & strId & conStrExt
        strFilename = Server.MapPath(strFilename)

        '** Create the local file (or append as appropriate)
        '**
        '** If you write directly to a database, this is where you need to put your code

        Set objFileSystem = Server.CreateObject("Scripting.FileSystemObject")

        If strSeq = conStrFirst Then
                Set objFile = objFileSystem.CreateTextFile(strFilename, true)
        Else
                Set objFile = objFileSystem.OpenTextFile(strFilename, conIntForAppending)
        End If

        If Not (IsNull(objFile) Or IsEmpty(objFile)) Then
                objFile.Write(strDecodedText)
                objFile.Close()
                Set objFile = nothing
                Set objFileSystem = nothing
        Else
                '** We seem to have a problem, so inform ViewONE

                strSuccess = "<FAILED>"
        End If

        If (Err) Then strSuccess = "<FAILED> " & Err.Description

        Response.Write vbLf & strSuccess


'*******************************************************
'* getParams()                                         *
'*******************************************************

Sub getParams()
        Dim arrStrPairs
        Dim strPair
        Dim strName
        Dim strValue
        Dim intPos

        strPair         = ""
        strName         = ""
        strValue        = ""
        intPos          = 0

        arrStrPairs = split(Request.QueryString, conStrParamDelim)

        For Each strPair In arrStrPairs
                intPos = InStr(strPair, conStrValDelim)
                If intPos > 1 Then
                        strName = Mid(strPair, 1, intPos-1)
                        strValue = Mid(strPair, intPos+1)

                        Select Case LCase(strName)
                                Case conStrSeq strSeq = strValue
                                Case conStrOf strOf = strValue
                                Case conStrName strName = strValue
                                Case conStrText strText = strValue
                                Case conStrId strId = strValue
                        End Select
                End If
        Next
End Sub
```

```
'*******************************************************
'* convertText()                                       *
'*******************************************************

Sub convertText()
        Dim strChar
        Dim i

        strChar = ""

        For i=1 To Len(strText) Step 2
                strChar = Mid(strText, i, 2)
                strDecodedText = strDecodedText & Chr(CInt("&H" & strChar))
        Next
End Sub
%>
```

Appendix F: Wang Compliant Annotation Types

COMPLIANT	NOT COMPLIANT
Line Text Solid Text Highlight Rectangle Square Redaction Line Text Stamp Image Stamp Ruler Angle	Arrow Note Hyperlink Polygon Open Polygon Redaction Polygon Oval Circle

Hyperlinks for all viewONE annotation types are NOT supported.

Ruler and Angle ViewONE annotations will not display as a ruler and angle in a Wang annotation viewer, e.g. Kodak Imaging, but will instead be represented as a line and two lines respectively.