IBM Daeja ViewONE Standard
IBM Daeja ViewONE Professional
IBM Daeja ViewONE Virtual

JavaScript Reference Manual

Version 4.1

SC27-6387-00

IBM Daeja ViewONE Standard
IBM Daeja ViewONE Professional
IBM Daeja ViewONE Virtual

JavaScript Reference Manual

Version 4.1

SC27-6387-00

Note
Before using this information and the product it supports, read the information in "Notices" on page 139.

This edition applies to Version 4 Release 1 Modification 0 of IBM® Daeja ViewONE Standard Edition (product number 5725-Q02), IBM Daeja ViewONE Professional Edition (product number 5725-Q03), and IBM Daeja ViewONE Virtual (product number 5725-Q04) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

ibm.com and related resources

Product support and documentation are available from ibm.com®.

Support and assistance

Product support is available on the web. Click Support from the product website at:

IBM Daeja ViewONE

http://www.ibm.com/support/entry/portal/product/enterprise_content_management/daeja_viewone

PDF publication

You can view the PDF files online by using the Adobe Acrobat Reader for your operating system. If you do not have the Acrobat Reader installed, you can download it from the Adobe website at http://www.adobe.com.

See the following PDF publication website:

Product	Website
IBM Daeja ViewONE Standard IBM Daeja ViewONE Professional IBM Daeja ViewONE Virtual	http://www.ibm.com/support/docview.wss?uid=swg27041437

How to send your comments

Your feedback helps IBM to provide quality information. Please share any comments that you have about this information or other documentation that IBM Software Development ships with its products.

You can use any of the following methods to provide comments:

- Send your comments by using the online readers' comment form at http://www.ibm.com/software/data/rcf/.

- Send your comments by email to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and publication number of the information (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Contacting IBM

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

To learn about available service options, call one of the following numbers:

- In the United States: 1-888-426-4343
- In Canada: 1-800-465-9600

For more information about how to contact IBM, see the Contact IBM website at http://www.ibm.com/contact/us/.

Introduction

IBM® Daeja® ViewONE® is a web-based image viewer 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.

It is provided as a modular product that allows you to extend this functionality to include additional format support, annotations (mark ups), and server-side components designed for performance optimization and additional functionality.

This document is the *IBM Daeja ViewONE JavaScript Reference Manual* and lists all the JavaScript methods available for configuring the viewer and is primarily designed for integrators of the product.

For further information about IBM Daeja ViewONE, consult the documentation available to download from here:

http://www.ibm.com/support/docview.wss?uid=swg27041437

IBM Daeja ViewONE JavaScript Overview

The IBM Daeja ViewONE JavaScript API offers an alternative mechanism to configure and control the IBM Daeja ViewONE applet.

In many cases the use of the JavaScript API is not required because most configuration parameters can be implemented using simple HTML (see *IBM Daeja ViewONE Parameters Reference Manual*).

However, where a more dynamic operation is required, for example, where it is wanted to configuration parameters during the operation of IBM Daeja ViewONE, perhaps to change a document without reloading a web page, rotating a page, and so on, then the JavaScript API is ideal.

Additionally, traditional web viewing applications habitually reload a web page to view or change the currently viewed image/document.

But with IBM Daeja ViewONE's extensive Image/Document open/close JavaScript API methods it is possible to avoid web page refreshing entirely. This can result in significant performance improvements since all delays due to web page refreshing (and in the case of the applet, therefore Java engine reloading and applet restarting) are avoided.

IBM Daeja has produced white paper specifically addressing performance tuning which covers the area of web page refreshing and applet reloading. You can access this white paper at: http://www.ibm.com/common/ssi/cgi-bin/ssialias?subtype=WH&infotype=SA&appname=SWGE_ZZ_VH_USEN&htmlfid=ZZW03282USEN&attachment=ZZW03282USEN.PDF.

Browser Note

Some browsers are case-sensitive. Therefore, you must enter method names as specified in this manual. Failure to adhere to this can result in methods not being called correctly.

JavaScript Reference

The JavaScript examples in this manual do not refer to their use in any particular context. The examples might be used within functions of a JavaScript program or directly as event handlers to buttons, hyperlinks, and so on.

File names and hyperlink addresses are expressed using the Internet URL address format (Uniform Resource Locator), for example "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 file names and hyperlink addresses, all parameters are case insensitive.

Security Note

Some JavaScript methods are disabled by default and can only be enabled by setting the "JavaScriptExtensions" HTML parameter set to **true** as follows:

<PARAM NAME="JavaScriptExtensions" VALUE="true">

Some JavaScript methods are disabled by default and can only be enabled by setting the "annotationJavaScriptExtensions" HTML parameter set to **true** as follows

<PARAM NAME="AnnotationJavaScriptExtensions" VALUE="true">

This is to prevent unauthorized users from attempting to manipulate IBM Daeja ViewONE through JavaScript methods to obtain access to secure information/documents/annotations.

The methods that are restricted in this manner are clearly marked within this document.

Opening documents and images

Method: **openFile(filename, page)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.openFile("myimages.tif", 1);

Specifies the file name and initial page of the document to be viewed.

The filename parameter can specify either the file name relative to the code base (as above) or the full URL. The code base is specified in the HTML code for the applet. An example of a full URL is as follows:

Example: ViewONE.openFile("http://mysite/myimages.tif", 1);

Method: **openDirectory(pathname)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.112 or later***
IBM Daeja ViewONE Professional V1.1.112 or later
No additional modules

Example: ViewONE.openDirectory("c:/images");

This parameter can be used to specify a local drive and directory. IBM Daeja ViewONE will search the path indicated by path name (including subdirectories) to locate all files. They will be opened, as one document (each file is assumed to be a single page image file).

This can be useful for cases where images are stored on hard disks, accessible locally (or by using a mapped drive) and the user wanted to view all files in that directory. You must always use forward-slashes as the path separator.

Method: **closeDocument()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.closeDocument();

Closes an open document.

NOTE: Closing the open document also resets annotations, annotation templates and background images.

Method Group:

initializePageArray(numPages)
setPageArray(filename, page)
openPageArray(page)

Requirements:

IBM Daeja ViewONE Standard, any version
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example:

```
ViewONE.initializePageArray(3);

ViewONE.setPageArray("page1.tif", 0);

ViewONE.setPageArray("page2.tif", 1);

ViewONE.setPageArray("page3.tif", 2);

ViewONE.openPageArray(1);
```

This method group specifies the number files (pages) in a list, then specifies each file (each one representing a successive page of the document), then opens the ‘assembled’ document at page 1. Note the page array begins at array element zero.

Initializing the page array causes a ‘soft’ close to be performed. That is, the current document is closed and the annotation source, templates and background images are reset if a document is currently open. If no document is currently open, then the close and reset is not performed.

Method Group:

initializePageAndThumbsArray(numPages)
setPageArray(filename, page)
setThumbsArray(filename, page)
openPageArray(page)

Requirements:

IBM Daeja ViewONE Standard, any version
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example:

```
ViewONE.initializePageAndThumbsArray(3);

ViewONE.setPageArray("page1.tif", 0);

ViewONE.setThumbsArray("page1-t.tif", 0);

ViewONE.setPageArray("page2.tif", 1);

ViewONE.setThumbsArray("page2-t.tif", 1);

ViewONE.setPageArray("page3.tif", 2);

ViewONE.setThumbsArray("page3-t.tif", 2);

ViewONE.openPageArray(1);
```

These methods are similar to the previous. They specify the number files (pages) in a list, and then specify a separate file for each page and a thumbnail file for that page. The final method then opens the "assembled" document at page 1. Note, the page array begins at array element zero.

In some instances, it may be advantageous to have separate files for the thumbnails to assist in browsing of thumbnails (smaller files are quicker to download and view).

Initializing the page and thumb arrays causes a ‘soft’ close to be performed. That is, the current document is closed and the annotation source, templates and background images are reset if a document is currently open. If no document is currently open, then the close and reset is not performed.

Method Group:

initializeDocumentArray(numDocs)
setDocumentArray(filename, index)
openDocumentArray()

Requirements:

IBM Daeja ViewONE Standard, any version
IBM Daeja ViewONE Professional, any version
No additional modules

Example:

```
ViewONE.initializeDocumentArray(3);
ViewONE.setDocumentArray("doc1parameters.txt", 0);
ViewONE.setDocumentArray ("doc2parameters.txt", 1);
ViewONE.setDocumentArray ("doc3parameters.txt", 2);
ViewONE.openDocumentArray(1);
```

This method group specifies the number of documents to open in a multi-document session, then specifies the parameter file for each document (each one representing a successive document), then opens the 'assembled' document list at document 1. Note the document array begins at array element zero.

For information about using document parameter files, see the *IBM Daeja ViewONE Parameters Reference Manual* for the section on the "Doc<N>" HTML parameter.

Initializing the document array causes a 'soft' close to be performed. That is, the current document is closed and the annotation source, templates and background images are reset if a document is currently open. If no document is currently open, then the close and reset is not performed.

Method:

openList(listFile, page)

Requirements:

IBM Daeja ViewONE Standard, any version
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.openList("mylist/list.txt", 1);

This method offers and alternative option to the page array methods above. The listFile parameter allows a file to be supplied which contains a list of pages.

This is useful for very large documents because it removes the need to deal with an array in JavaScript.

It can also be used to keep the HTML constant, by changing the source list instead of changing the HTML between different documents.

Method: **reloadList()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.reloadList()

This method forces the list used with the openList() method to be reloaded and the document to be re-opened. It will reload the list file from source (that is, the web server) each time, so if it has changed then the changes will be picked up.

Method: **openDoc(index)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.openDoc(2)

Opens the document that is specified by the index parameter.

This method applies only when multiple documents are loaded into the viewer using the "doc<N>" HTML parameter (see the *IBM Daeja ViewONE Parameters Reference Manual*) or the Document Array JavaScript equivalent.

The value of "index" represents the associated "doc<N>" parameter. Therefore, the above example will cause IBM Daeja ViewONE to open the second document in the list (that is, that specified by the "doc2" HTML parameter).

Method: **nextDoc()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.nextDoc()

This method applies only when multiple documents are loaded into the viewer using the "doc<N>" HTML parameter (see IBM Daeja ViewONE Parameters Reference Manual) or the Document Array JavaScript equivalent.

Calling this method is the JavaScript equivalent of clicking on the Next Document button on the top toolbar when multiple documents are loaded in the viewer. If it is called when on the last document in the list it has no effect.

Method: **prevDoc()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.prevDoc()

This method applies only when multiple documents are loaded into the viewer using the "doc<N>" HTML parameter (see IBM Daeja ViewONE Parameters Reference Manual) or the Document Array JavaScript equivalent.

Calling this method is the JavaScript equivalent of clicking on the Previous Document button on the top toolbar when multiple documents are loaded in the viewer. If it is called when on the first document in the list it has no effect.

Method: **firstDoc()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.firstDoc()

This method applies only when multiple documents are loaded into the viewer using the "doc<N>" HTML parameter (see IBM Daeja ViewONE Parameters Reference Manual) or the Document Array JavaScript equivalent.

Calling this method is the JavaScript equivalent of clicking on the First Document button on the top toolbar when multiple documents are loaded in the viewer. If it is called when on the first document in the list it has no effect.

Method: **lastDoc()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.lastDoc()

This method applies only when multiple documents are loaded into the viewer using the "doc<N>" HTML parameter (see IBM Daeja ViewONE Parameters Reference Manual) or the Document Array JavaScript equivalent.

Calling this method is the JavaScript equivalent of clicking on the Last Document button on the top toolbar when multiple documents are loaded in the viewer. If it is called when on the last document in the list it has no effect.

Method: **getDocIndex()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var currentdoc = ViewONE.getDocIndex()

This method returns the index value of the currently viewed document when multiple documents are loaded into the viewer using the "doc<N>" HTML parameter (see IBM Daeja ViewONE Parameters Reference Manual) or the Document Array JavaScript equivalent.

Note: The index starts at "1" which represents the first document in the list.

Method: **getNumDocs()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var numdocs = ViewONE.getNumDocs()

This method returns an integer which represents the total number of documents loaded into the viewer using the "doc<N>" HTML parameter (see IBM Daeja ViewONE Parameters Reference Manual) or the Document Array JavaScript equivalent.

Method: **getNumPages()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var numPages = ViewONE.getNumPages();

Gets the number of pages in the current document (an integer).

Method: **setPage(page)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setPage(2);

Sets the current page number (an integer).

Method: **getPage()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var page = ViewONE.getPage();

Returns the current page number as an integer.

Method: **nextPage()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.nextPage();

Convenience method to view the next page (current page + 1)

Method: **previousPage()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.previousPage();

Convenience method to view the previous page (current page - 1).

Method Group: **setDescription(description)**
getDescription()

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example::

```
var desc = ViewONE.getDescription();
ViewONE.setDescription("myDoc");
```

Gets or sets the description for the document. The description is used when opening the document in IBM Daeja ViewONE, "Opening docXYZ…" or "Opening docXYZ, page 2…." will be displayed in the status bar.

Method: **getDocReference()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example:

```
var ref = ViewONE.getDocReference();
```

Gets the document reference for the document.

Method: **showExternalEmailImages(boolean)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

When called the email will be redisplayed with the external images visible (true) or blocked (false). This method requires that the HTML parameter for 'allowExternalEmailImages' is set to true (see IBM Daeja ViewONE Parameters Reference Manual for details).

Method: **setBackgroundImageEnabled(boolean)**

Requirements: ***IBM Daeja ViewONE Professional, any version***
No additional modules

This method is only applicable when used with the COLD (Professional only) module and where a background image has been specified. When set to true, the background image will be visible. A setting of false removes the background image from the viewer.

Method: **setDocumentIndexList()**

Requirements: ***IBM Daeja ViewONE Standard V3.0.490 or later***
IBM Daeja ViewONE Professional V1.0.490 or later
No additional modules

Example: ViewONE.setDocumentIndexListFile("mylist.lst", 1);

Updates the text based index list of documents.

This method allows you to dynamically specify a different file that contains the replacement text. The file must be a simple text file, with <LF> delimiting successive document indexes.

Method: **setDocumentCookies (String value)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.172 or later***
IBM Daeja ViewONE Professional V1.1.172 or later
No additional modules

Use this method to set the cookies that will be used for the next document request from the document server. It is useful when using the viewer in an Ajax style environment when a user may have logged in to a resource remotely without a refresh of the current viewer HTML page.

The cookies should be supplied in a semi-colon separated list, for example:

ViewONE.setDocumentCookies("mysessionid=29;userid=abc");

The format of the document.cookie JavaScript property is also supported:

ViewONE.setDocumentCookies(document.cookie);

Specifying an Annotations File

Method: **setAnnotationFile(filename)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setAnnotationFile("http://mysite/myannotations.ant");

Sets the annotation file and must be called before any of the open methods described in the preceding section. This method requires that annotationJavaScriptExtensions HTML parameter is set to true.

If the current document is closed after this method has been called, the annotation file will be reset. To avoid this, close the current document before setting the annotation file, as closing of the current document can be a side effect of calling other JavaScript methods, for example, 'initializePageArray'.

Examples:

viewONE.setAnnotationFile("http://...annotations.ant");

viewONE.openFile("http://...mydocument.tif", 1);

or

```
viewONE.closeDocument();
viewONE.setAnnotationFile("http://...annotations.ant");
viewONE.initializePageArray(2);
viewONE.setPageArray("http://....page1.tiff", 0);
viewONE.setPageArray("http://....page2.tif", 1);
viewONE.openPageArray(1);
```

Method: **setShowAnnotations(true/false)**

Requirements: ***IBM Daeja ViewONE Standard V3.0.598 or later***
IBM Daeja ViewONE Professional V1.0.598 or later
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setShowAnnotations(false);

This method changes whether IBM Daeja ViewONEdisplays annotations (false = no, true = yes). Note: that if "false" is selected then IBM Daeja ViewONEwill only hide those annotations which the user is able to hide (that is, those annotations for which the user has edit privileges).

Method: **isShowAnnotations()**

Requirements: ***IBM Daeja ViewONE Standard V3.0.598 or later***
IBM Daeja ViewONE Professional V1.0.598 or later
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: if (ViewONE.isShowAnnotations()) …

This method returns ‘true’ if annotations are allowed to be displayed, ‘false’ if not.

Saving documents and images

Method: **save(filename)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.save("c:/temp/image.tif");

Saves the current document (multi-page tiffs), or current page (multi-file documents) to the specified filename.

If filename is null, either the save dialog is displayed or, if a server side save component has been specified, the image is sent to the server-side component.

Note: The save operation does not re-create the source file, it merely copies the source file to the specified destination and renames it to the specified filename.

Method: **saveAsFormat(filename, format)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.210 or later***
IBM Daeja ViewONE Professional V1.1.210 or later
NO additional modules

Example: ViewONE.saveAsFormat("c:/temp/page.tif", “tiff”);

Saves the current document, with the specified format.

If filename is null, either the save dialog is displayed or, if a server side save component has been specified, the image is sent to the server-side component.

The only format currently supported is “tiff”.

If the format specified is null, and the document is a single file, multipage document, then the source file will be copied to the destination.

If the format specified is “tiff” and the document is not a single file, multipage TIFF, then a new TIFF file will be created.

Method: **savePage(filename)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.savePage("c:/temp/page.tif");

Saves the current page to the specified filename.

If filename is null, either the save dialog is displayed or, if a server side save component has been specified, the image is sent to the server-side component.

If the document is a multi-page TIFF then this method will extract the current page from the source TIFF file, and create a new TIFF file containing only the current page.

Method: **saveSelected(filename)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.saveSelected("c:/temp/selected.tif");

Saves the current selected pages of a multi-page TIFF document.

If filename is null, either the save dialog is displayed or, if a server side save component has been specified, the image is sent to the server-side component.

Note: this method is for use only with multi-page TIFF documents. If you are viewing a multi-file document then it is not possible to save selected pages from that document except by individually calling the savePage() method for each page that you want to save. You may find isMultipageTif() and getSelection() methods useful to call before calling the saveSelected method.

Method: **isMultipageTif()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var ismulti = ViewONE.isMultipageTif();

This method is useful if you want to first check whether the current document is a multi-page TIFF before using any of the Save methods. If it is then this method returns true else it returns false.

Method: **setImageSaveURL(saveURL, type, useButton)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setImageSaveURL("http://myServer/imageSave", 2, true);

This method is used to set the URL to be used for image saving.

The URL is specified using the saveURL parameter.

The type of saving mechanism is specified with the type parameter. The available types are as follows:

0 - Save using GET method (now deprecated).
1 – Save using the POST method.
2 – Save using the SERVLET method.

If the useButton parameter is set to true, the save button uses the specified URL If this parameter is set to false, only embedded annotation saving uses the URL.

Method: **setDefaultSaveFilename(filename)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.96 or later***
IBM Daeja ViewONE Professional V1.1.96 or later
No additional modules

Example: ViewONE. setDefaultSaveFilename ("myFile.tif");

Specifies the file name that appears in the save dialogs file name field every time the save dialog appears. This is a document-specific parameter and must be specified for each instance of ViewONE.

Document hyperlinks

Method: **setHyperlink(url, dblClick)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setHyperlink("newpage.html", false);

or

ViewONE.setHyperlink("http://mysite/newpage.html", false);

Specifies a hyperlink that is activated when the user clicks on the image area. If the dblClick parameter is 'true' then the hyperlink is activated only after the user double-clicks on the image area, otherwise it requires only a single click.

The hyperlink can specify either the filename relative to the code base or the full URL as illustrated.

Method: **clearHyperlink()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.clearHyperlink();

This method clears a hyperlink if one has previously been defined using the setHyperlink() method.

Image

Method: **invert()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
ViewONE Virtual
No additional modules

Example: ViewONE.invert();

Inverts the display colors (black changes to white and vice versa). This method is also effective on images with more than two colors. A second call to this method will re-establish the original display colors.

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **setInverted(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setInverted(true);

Sets the invert of the display colors (black changes to white and vice versa). This method is also effective on images with more than two colors

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **isInverted()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var inverted = ViewONE.isInverted();

Returns a Boolean ‘true’ if the colors are inverted, ‘false’ if they are not.

Method: **setEnhance(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var ViewONE.setEnhance(true);

Specifies whether a monochrome image is displayed with anti-aliasing on or off.

Method: **isEnhance(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var enhance = ViewONE.isEnhance();

Returns a Boolean ‘true’ if enhance is on, ‘false’ if it is off.

Method:

setEnhanceMode(mode)

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setEnhanceMode(1);

Specifies which anti-aliasing algorithm to use:

0 -Enhance off
1 - Enhance level (typical images)
2 - Enhance level (drawings)
3 - Enhance level (very high-resolution drawings)

Note: This method can be used to turn on/off Anti-Aliasing and so there is no need to use both this and setEnhance methods together.

Method: **getEnhanceMode()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var state = ViewONE.getEnhanceMode();

Returns the current enhance state.

Method: **setRotation(angle)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setRotation(90);

Specifies the angle at which pages are displayed. Values of 90, 180, or 270 are accepted. The default is 0.

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **initializeRotationArray(int size)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.initializeRotationArray(4);

Initializes the rotation array, used for setting the rotation of each page of the current image, to the given size.

Method: **setRotationArray(int angle, int page)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var ok = ViewONE.setRotationArray(90, 0);

Sets the rotation for given page to the given angle. The pages are indexed from 0, so the first page is zero, and the second is 1.

The specified angle, which is in degrees, must be set to 0, 90, 180 or 270.

The JavaScript method initializeRotationArray() should be used to set the size of the rotation array before this method is called.

The method returns true if the page rotation was successfully set or false if the set failed, for instance if the specified page is an invalid index in the rotation array initialized using initializeRotationArray().

Method: **applyRotationArray()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.applyRotationArray();

Applies the rotation array that was set up by using the initializeRotationArray() and setRotationArray() methods to the current image.

Method: **getRotation()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var angle = ViewONE.getRotation();

Returns the angle of rotation as an integer.

Method: **rotateClockwise()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.rotateClockwise();

Convenience method to increase the rotation by 90 degrees.

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **rotateCounterclockwise()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
ViewONE Professional /ViewONE Virtual
No additional modules

Example: ViewONE.rotateCounterclockwise();

Convenience method to decrease the rotation by 90 degrees.

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **rotate180()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.rotate180();

Convenience method to rotate the document to 180 degrees.

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **setFlip(mode)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setFlip(1);

Specifies the flip mode of displayed pages. Values of 0 (none), 1 (horizontal or mirror), 2 (vertical) or 3 (both) are accepted. The default is 0.

Flip buttons and menus are not visible to the user by default. To enable these options for the user you must use the flipOptions HTML tag when the applet is loaded.

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **getFlip()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var angle = ViewONE.getFlip();

Returns the flip mode as an integer.

Method: **setScale(scale)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setScale(0);

Specifies the scale mode used to display a page. Acceptable integer values are:

0: best fit

The page is scaled to fit into the window area so that the entire page is visible.

1: Fit-to-window-width

The page is scaled so that the width of the page matches the width of the window area. This may result in the visible page height exceeding the available window height in which case a vertical scroll bar appears automatically.

2: Fit-to-window-height

The page is scaled so that the height of the page matches the height of the window area. This may result in the visible page width exceeding the available window width in which case a horizontal scroll bar appears

Method: **getScale()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var scale = ViewONE.getScale();

Returns the integer scale value.

Method: **getStates()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: states = ViewONE.getStates();

Returns a coded string to be used with setStates(). When this method is called while a document is open, it returns a string containing information about the current zoom, scroll and other states.

Method: **setStates(string states)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setStates(states);

Sets the zoom, scroll and other states to the values specified by the coded string. This method should be called before opening a document.

The getStates() and setStates() methods together permit the viewing states to be restored when a document is closed and re-opened.

Method: **zoomIn()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.zoomIn();

Applies a 25% increase in zoom. Note, at first ViewONE will attempt to use the scale modes (fit-to-width, fit-to-height and best-fit) if they are more appropriate. The zoomIn() function will increase the zoom factor only after the scale modes are no longer suitable.

Method: **zoomOut()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.zoomOut();

Reverses the effect of zoomIn().

Method: **zoom100()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.zoom100();

Zooms image to 100% (full resolution).

Method: **setZoom(zoom)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setZoom(2.0);

Zooms to the value specified. The value is specified as a double value - 1.0 = 100% or 1 image pixel = 1 screen pixel, 2.0 = 200% or 1 image pixel = 2 screen pixels.

If used in conjunction with setXYScroll(), it is advisable to use setZoomAndXYScroll() instead as problems may occur when both setZoom() and setXYScroll() are used together.

Method: **getZoom()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var zoom = ViewONE.getZoom();

Returns the current zoom level as a double value - 1.0 = 100% or 1 image pixel = 1 screen pixel, 2.0 = 200% or 1 image pixel = 2 screen pixels.

Method: **setXYScroll(x, y)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setXYScroll(20,20);

Sets the scroll bar positions to the given values in screen pixels.

If setZoom() is used in conjunction with setXYScroll(), it is advisable to use setZoomAndXYScroll() instead as problems may occur when both setZoom() and setXYScroll() are used together.

Method: **getXScroll()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var xScroll = ViewONE.getXScroll();

Returns the position of the horizontal scrollbar as a percentage of the entire scrollbar, where furthest left is 0 and furthest right is 100.

Method: **getYScroll()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var yScroll = ViewONE.getYScroll();

Returns the position of the vertical scrollbar as a percentage of the entire scrollbar, where top is 0 and bottom is 100.

Method: **setZoomAndXYScroll(zoom, x, y)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setZoomAndXYScroll(2.0, 100, 100);

Zooms to the value specified. The value is specified as a double value - 1.0 = 100% or 1 image pixel = 1 screen pixel, 2.0 = 200% or 1 image pixel = 2 screen pixels.

Method: **zoomArea(x, y, width, height, highlight, seconds)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.zoomArea(400, 500, 100, 150, true, 2);

Zooms to an area starting at "x", "y" which is "width" across and "height" high. If "highlight" is set "true" then the area zoomed is also highlighted for a time specified by the "seconds" parameter.

If "seconds" is greater than 0 then the highlight is visible for that time. If "seconds" is less than or equal to 0 then the highlight will remain visible until the user clicks or forces a refresh by scrolling, rotating, or so on.

x, y, width, height = integers specifying image pixel values
seconds = integer specifying seconds
highlight = boolean

If the area specified is a different aspect ratio from the display area then the viewer will attempt to fit the zoom as best it can.

Method: **setDraggingEnabled(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setDraggingEnabled(true);

Specifies whether the dragging of the image is permitted or not (using the mouse). Dragging the image to the right pans the image to the right, dragging the image to the left pans the image to the left, and so on.

A value of 'true' (default) indicates dragging is permitted and 'false' indicates that it is not

Method: **isDraggingEnabled()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var draggingEnabled = ViewONE.isDraggingEnabled();

Returns a Boolean value of 'true' if dragging is allowed else a value 'false' is returned.

Method: **setBrightness(percent)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setBrighness(60);

This method sets the brightness of the display. The value represents a percentage from 0-100, with 50 being the default value. 0 = minimum brightness (dark) and 100 = maximum brightness (light).

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **resetBrightness()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.resetBrighness();

This method resets the brightness level to 50% (the default value).

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **getBrightness()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var percent = ViewONE.getBrighness();

Returns the current brightness percentage setting (0-100).

Method: **setContrast(percent)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setContrast(40);

This method sets the contrast of the image displayed. The value represents a percentage from 0-100, with 50 being the default value. 0 = minimum contrast (flat) and 100 = maximum brightness (not flat!).

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **resetContrast()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.resetContrast ();

This method resets the contrast level to 50% (the default value).

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **getContrast()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var percent = ViewONE.getContrast();

Returns the current contrast percentage setting (0-100).

Method: **setLuminance(percent)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setLuminance(70);

This method sets the luminance of the image displayed. The value represents a percentage from 0-100, with 50 being the default value. 0 = minimum luminance (dull) and 100 = maximum brightness (bright).

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Brightness increases the brightness of all colors (dark and light) uniformly, whereas luminance causes already bright areas of the image to increase in brightness further and darker areas to increase in brightness also, but by a lesser amount. This simulates a light source shining on the image and can be more effective at making color images clearer to read.

Method: **resetLuminance()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.resetLuminance();

This method resets the luminance level to 50% (the default value).

If the document is closed when this method is called, the default for all pages is modified. If the document is open when this method is called, the individual page is modified only.

Method: **getLuminance()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var percent = ViewONE.getLuminance();

Returns the current luminance percentage setting (0-100).

Method: **getImageWidth()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var width = ViewONE.getImageWidth();

Returns an integer value representing the width of the currently displayed image in image pixels.

Method: **getImageHeight()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var height = ViewONE.getImageHeight();

Returns an integer value representing the height of the currently displayed image in image pixels.

Method: **getXResolution()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var xRes = ViewONE.getXResolution();

Returns an integer value representing the x-axis resolution of the currently displayed image in dots per inch.

The value is obtained from the image's header information, so if the information is missing or corrupt the returned value will make no sense.

Method: **getYResolution()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var yRes = ViewONE.getYResolution();

Returns an integer value representing the y-axis resolution of the currently displayed image in dots per image.

The value is obtained from the image's header information, so if the information is missing or corrupt the returned value will make no sense.

Viewing

Method: **setView(view)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setView(0);

Specifies the view mode used to display pages of a document. This method is effective only while a document is open. Acceptable integer values are:

0: Fullpage

(default): A single view of the current page is visible

1: Twopage

Two pages are visible at the same time

2: Thumbsonly

A view of the thumbnails only is visible.

3: Thumbsleft

A view of the current page with thumbnails on the left of the page.

4: Thumbsright

A view of the current page with thumbnails on the right of the page.

5: Thumbstop

A view of the current page with thumbnails at the top of the page.

6: Thumbsbottom

A view of the current page with thumbnails at the bottom of the page.

Method: **getView()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var scale = ViewONE.getView();

Returns the integer view mode value.

Method: **setAreaZoom(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setAreaZoom (true);

If true, initiates the zoom-area mode. The mouse pointer changes to a cross and the user can drag the mouse (using button one) to select an area for zooming. When the mouse button is released the area selected will be zoomed as large as possible whilst maintaining the image aspect within the available window area.

If the selected area is not greater than the zoom trigger size (currently 20*20 pixels) then zooming will not occur. This allows the user to release the mouse if the mode was initiated accidentally.

If false, mouse functionality returns to drag mode (to pan the image).

Method: **isAreaZoom()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var areaZoom = ViewONE.isAreaZoom();

Returns a Boolean value indicating the zoom-area status.

Method: **toggleAreaZoom()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.toggleAreaZoom();

This toggles IBM Daeja ViewONE zoom area mode. When in zoom area mode the cursor will change to a cross hair and the user can select an area to zoom using the cursor. When not in zoom area mode, the cursor can be used to drag or scroll the image.

Method: **setMagnifier(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setMagnifier(true);

If true, displays an external magnifier window. A rectangle is visible around the mouse pointer which highlights the area being magnified.

If false, the magnifier window is hidden.

Method: **setMagnifierInternal(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setMagnifierInternal(true);

If true, displays a magnifier window internal to the display area.

If false, the magnifier window is hidden.

Method: **isMagnifier()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var magVisible = ViewONE.isMagnifier();

Returns a Boolean value indicating the magnifier visibility status.

Method: **toggleMagnifier()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.toggleMagnifier();

Toggles the magnifier on/off.

Method: **setMagFactor()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setMagFactor(int factor);

Sets the current integer magnification factor (for magnifier window)

Method: **getMagFactor()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var factor = ViewONE.getMagFactor();

Returns the current integer magnification factor (for magnifier window)

Method: **setMagBounds(int x, int y, int width, int height)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setMagBounds(10, 10, 300, 300);

Sets the external magnifier's window position and size.

Method: **setNewWindowVisible(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setNewWindowVisible(true);

Specifies whether to make the IBM Daeja ViewONE new window visible. A value of 'true' makes the window visible and 'false' (default) makes it invisible.

Method: **isNewWindowVisible()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setNewWindowVisible(!ViewONE.isNewWindowVisible());

Returns a Boolean value of 'true' if the IBM Daeja ViewONE new window is visible else a value 'false' is returned.

Method: **setImageForeColor(color)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setImageForeColor("0, 0, 0")

This method sets the default color used for the foreground (text) of monochrome images.

Colors are specified in a string format using the standard RGB values.

Method: **showImageForeColorDialog()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.showImageForeColorDialog();

This method will cause IBM Daeja ViewONE to display a dialog to allow the user to change the default color used for the foreground (text) of monochrome images.

Method: **setImageBackColor(color)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setImageBackColor("255, 255, 255")

This method sets the default color used for the background of monochrome images.

Colors are specified in a string format using the standard RGB values.

Method: **showImageBackColorDialog()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.showImageBackColorDialog();

This method will cause IBM Daeja ViewONE to display a dialog to allow the user to change the default color used for the background of monochrome images.

Labels

Method Group:

initializeLabels(numLabels)
setLabel(pageLabel, pageLabelColor, thumbLabel, thumbLabelColor, labelNum)
useLabels()
clearLabels()

Requirements:

IBM Daeja ViewONE Standard, any version
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example:

```
ViewONE.initializeLabels(3);

ViewONE.setLabel("Page label 1", null, "thumb 1", null, 0);

ViewONE.setLabel("Page label 2", null, "thumb 2", null, 1);

ViewONE.setLabel("Page label 3", "223,223,255", "thumb 3", "255,223,223", 2);

ViewONE.useLabels();

ViewONE.openFile("mydocument.tif", 1);
```

This method group specifies the number files (labels) in a list, then specifies each label. Each one representing a successive page of the document and specifying the label to be displayed for the full-page area and the corresponding thumbnail. It then sets the labels in use by calling the useLabels() method.

Label colors are specified using the standard RGB values and where no color is specified (that is, null) then the default color is white.

Notes:

The label array begins at array element zero.

Labels will remain visible until the document is closed or the clearLabels() method is called.

If you do not want to define a label for either the full-page area or a thumbnail then specify the label as a null string, for example:

```
ViewONE.setLabel(null, null, "thumb 1", null, 0);
```

This example sets up a label for the thumbnail for the first page.

Selection and clipboard

Method: **selectPage(int pageNumber)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.selectPage(4);

Toggles the select property on the page in a document indicated by the "pageNumber" parameter or getPage() JavaScript method (available for multi-page documents only).

Method: **clearSelections()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.clearSelections();

Clears all page selections in the document (available for multi-page documents only).

Method: **copyPageToClipboard()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.copyPageToClipboard();

This method takes a copy of the currently selected page and keeps it in the Windows clipboard for pasting into other applications.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **startCopyAreaToClipboard()**

Requirements: ***IBM Daeja ViewONE Standard V4.0.34 or later***
IBM Daeja ViewONE Professional V4.0.34 or later
No additional modules

Example: ViewONE.startCopyAreaToClipboard();

This method replicates the same behavior as if the user had clicked on the “Copy selected area to clipboard” function from the right-click “Clipboard” menu in the viewer.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **getSelection()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: selection = ViewONE.getSelection();

This method returns a comma-delimited string containing any pages selected by the user.

Printing

Method: **printPage()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.printPage();

Produces a print dialog to allow the user to print the current page.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **printDocument()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.printDocument();

Produces a print dialog to allow the user to print the current document (available for multi-page documents only).

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **printRange()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.printRange();

Produces a range dialog which to allows the user to print a range of pages in a document (available for multi-page documents only).

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **printSelected()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.printSelected();

Produces a print dialog to print pages selected using the page-select menu (available for multi-page documents only). Can be used with the "selectPage(pageNumber)" and "clearSelections()" methods to print any page or group of pages within a document.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **printVisible()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.printVisible();

Produces a print dialog to print the image display (visible).

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **printTransformed()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Print Accelerator module

Example: ViewONE.printTransformed();

Generates a print with rotate, invert and flip modes applied to the image.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **setPrintDialog(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Print Accelerator module

Example: ViewONE.setPrintDialog(false);

This method applies only when the print-accelerator is used. When this parameter is set to false, printing will take place without showing the standard print-dialog. If the user has not printed using IBM Daeja ViewONE with the accelerator previously, then the users' default printer will be used. Otherwise, the last printer used by the user (with IBM Daeja ViewONE print accelerator) will be used.

Method: **isPrintDialog()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Print Accelerator module

Example: var dialog = ViewONE.isPrintDialog();

Returns a Boolean 'True' if the print dialog is enabled, 'False' if it is not.

Method: **setPrintCopies(integer)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Print Accelerator module

Example: ViewONE.setPrintCopies(2);

This method is effective only when the print dialog is disabled (by calling setPrintDialog(false) or using the equivalent HTML tag). This method sets the number of copies that will be printed when printing a page, pages or the document.

Method: **setPrinter(string)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Print Accelerator module

Example: ViewONE.setPrinter("myprinter");

This method is effective only when the print dialog is disabled (by calling setPrintDialog(false) or using the equivalent HTML tag). This method sets the IBM Daeja ViewONE default printer to the one specified as the parameter. The parameter must be the 'name' (or unique part thereof) as seen by the users printer settings. The printers' default settings will be used for each print (for example, orientation and resolution).

Method: **setPrintHeader(headerString)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Print Accelerator module

Example: ViewONE.setPrintHeader("$page # $of ##")

Printouts can include custom text at the top of each page. By default this text is set to the page number followed by the number of pages in the document. The following options are available:

"false" This value will disable print headers

"any text" This is the text that will appear at the top of each printed page. For example, it might be your own copyright for the documents being viewed or some other informational text.

"formatted text" The text can include some limited formatted elements as follows:

$page : This will print the word "page" in the appropriate translation

$of : Similarly for the word "of"

$pages: Similarly for the word "pages"

: This will print the page number of the page being printed

: This will print the number of pages in the document

Example:

"$page # $of ## © Copyright blah 2000"

Note 1: The default value is: "($page # $of ##)"

Note 2: Print headers are available for Internet Explorer 4.01 or later, Netscape 4.06 or later, and the Java Plug-in 1.3. (The Java Plug-in 1.2.2 does not offer sufficient functionality to permit print headers and so they will not be seen if using this version of the plug-in).

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **setPrintAutoRotate (true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Print Accelerator module

Example: ViewONE.setPrintAutoRotate(true)

This parameter is only available applicable when the tag multiPrintNum is set to more than 1 (see IBM Daeja ViewONE Parameters Reference Manual for details).

If this parameter is set to true, then IBM Daeja ViewONE will attempt to rotate images automatically before printing so that as many images can be printed (vertically) on a printed page as possible. The default value for this tag is false.

Method: **setPrintJobName (string)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.144 or later***
IBM Daeja ViewONE Professional V1.1.144 or later
Print Accelerator module

Example: ViewONE.setPrintJobName("my print job")

This parameter sets the print job name sent to the printer when IBM Daeja ViewONE prints.

Note: The functionality provided by this parameter is only available when using the print accelerator module for IBM Daeja ViewONE or when the Oracle Java Plug-in v1.2 or later is in use.

Document text search

Document text search allows JavaScript control over IBM Daeja ViewONE Professional's text finding capability.

Note: Currently supported by the PDF module only. Other file formats cannot be searched. Trying to run any of the JavaScript methods listed in this section when other file formats are loaded will result in nothing happening.

Method: **find(String text, boolean caseSensitive, boolean wholeWord)**

Requirements: ***IBM Daeja ViewONE Professional V1.1.70 or later***
PDF module

Example: ViewONE.find("view", false, true);

This JavaScript method is used to start a document find. The "text" parameter should specify the term to find. If "caseSensitive "is set to true, matches will be case-sensitive. If "wholeWorld" is set to true, then partial word matches will not count. For example, if searching for "view", as above, the word "view" would match, but the word "ViewONE" would not.

Calling this method whilst a find is running (either started from the IBM Daeja ViewONE GUI or from JavaScript) will cause the existing find process to be canceled.

Method: **cancelFind()**

Requirements: ***IBM Daeja ViewONE Professional V1.1.70 or later***
PDF module

Example: ViewONE.cancelFind();

This JavaScript method will cancel any running find process including one that was started by the user from the IBM Daeja ViewONE GUI.

Method: **clearFindResults()**

Requirements: ***IBM Daeja ViewONE Professional V1.1.70 or later***
PDF module

Example: ViewONE.clearFindResults();

This JavaScript method will clear any search results shown in the thumbs tool pane and clear the search result highlights from the document viewing area.

Method: **nextFindResult()**

Requirements: ***IBM Daeja ViewONE Professional V1.1.70 or later***
PDF module

Example: ViewONE.nextFindResult ();

This JavaScript method will highlight the next find result in the viewer thumbnail pane's search view and also the document viewing area. If no find result is selected, the first find result will be highlighted. Highlighting a find result will also include changing to the page that the find result is on. It is equivalent to clicking the "Next result" button on the IBM Daeja ViewONE GUI (in the thumbnail pane's search view).

Method: **prevFindResults()**

Requirements: ***IBM Daeja ViewONE Professional V1.1.70 or later***
PDF module

Example: ViewONE.prevFindResult();

This JavaScript method will highlight the previous find result. Highlighting a find result will also include changing to the page that the find result is on. It is equivalent to clicking the "Previous result" button on the IBM Daeja ViewONE GUI (in the thumbnail pane's search view).

Toolbars and Buttons

Method: **setScrollbars(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setScrollbars(true);

Specifies whether scrollbars will appear when the image is scaled to a size larger than the display area. A value of 'true' (default) indicates scrollbars are required and 'false' indicates they are not. A change in this setting will be visible after the next refresh of the display (for example, when a page is zoomed or a page is changed).

Method: **isScrollbars()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var scrollbars = ViewONE.isScrollbars();

Returns a Boolean value of 'true' if scrollbars are enabled else a value 'false' is returned.

Method: **setStatusBar(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setStatusBar(true);

Specifies whether the status bar is visible or not. A value of 'true' (default) indicates the status bar is visible and 'false' indicates that it is not.

Method: **isStatusBar()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var statusBarVisible = ViewONE.isStatusBar();

Returns a Boolean value of 'true' if the status bar is visible else a value 'false' is returned.

Method: **setFileButtons(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setFileButtons(true);

Specifies whether the toolbar includes file buttons. A value of 'true' (default) indicates the buttons are visible and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isFileButtons()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var buttonsVisible = ViewONE.isFileButtons();

Returns a Boolean value of ‘true’ if the file buttons are visible else a value ‘false’ is returned.

Method: **setImageButtons(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setImageButtons(true);

Specifies whether the toolbar includes image buttons. A value of 'true' (default) indicates the buttons are visible and 'false' indicates they are not.

The image buttons are:

Zoom area, Zoom in, Zoom out, Fit to width, Fit to height, Best fit, Rotate clockwise, Rotate counterclockwise, Rotate 180.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isImageButtons()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var buttonsVisible = ViewONE.isImageButtons();

Returns a Boolean value of 'true' if the image buttons are visible else a value 'false' is returned.

Method: **setPrintButtons(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setPrintButtons(true);

Specifies whether the toolbar includes a print button. A value of 'true' (default) indicates the button is visible and 'false' indicates it is not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isPrintButtons()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var buttonVisible = ViewONE.isPrintButtons();

Returns a Boolean value of 'true' if the print button is visible else a value 'false' is returned.

Method: **setInvertButtons(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setInvertButtons(true);

Specifies whether the toolbar includes an invert button. A value of 'true' (default) indicates the button is visible and 'false' indicates it is not.

Method: **isInvertButtons()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var buttonVisible = ViewONE.isInvertButtons();

Returns a Boolean value of 'true' if the invert button is visible else a value 'false' is returned.

Method: **setNewWindowButtons(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setNewWindowButtons(true);

Specifies whether the toolbar includes a new-window button. A value of 'true' (default) indicates the button is visible and 'false' indicates it is not.

Method: **isNewWindowButtons()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var buttonVisible = ViewONE.isNewWindowButtons();

Returns a Boolean value of 'true' if the new-window button is visible else a value 'false' is returned.

Method: **setViewButtons(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setViewButtons(true);

Specifies whether the toolbar includes view buttons. A value of 'true' (default) indicates the buttons are visible and 'false' indicates they are not.

The view buttons are:

Fullpage, Thumbnails, Two-page, Thumbs-left, Thumbs-bottom, Thumbs-right, Thumbs-top.

Method: **isViewButtons()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var buttonsVisible = ViewONE.isViewButtons();

Returns a Boolean value of 'true' if the view buttons are visible else a value 'false' is returned.

Method: **setAllButtons(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version

IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setAllButtons(true);

Specifies whether all buttons in the top and bottom toolbars are visible or not. This includes the file, print, zoom, transformation and new window options in the top toolbar and the view options from the lower toolbar.

A value of 'true' (default) indicates the buttons are visible and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isAllButtons()**

Requirements:
IBM Daeja ViewONE Standard, any version
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var buttonsVisible = ViewONE.isAllButtons();

Returns a Boolean value of 'true' if the all buttons are visible else a value 'false' is returned (these are file, print, image, new-window and view buttons)

Method: **setPageButtons(true/false)**

Requirements:
IBM Daeja ViewONE Standard, any version
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: ViewONE.setPageButtons(true);

Specifies whether the toolbar includes page buttons. A value of 'true' (default) indicates the buttons are visible and 'false' indicates they are not.

The page buttons are:

First page, previous page, next page, last page and a drag bar for page selection..

Method: **isPageButtons()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
No additional modules

Example: var buttonsVisible = ViewONE.isPageButtons();

Returns a Boolean value of 'true' if the page buttons are visible else a value 'false' is returned.

Method: **toggleAdjustTool()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.toggleAdjustTool();

This method toggles the visibility of the adjust tool (brightness/contrast/luminance).

Method: **setAdjustToolVisible(OnOff)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setAdjustToolVisible(True);

This method sets the visibility of the adjust tool (brightness/contrast/luminance).

Method: **isAdjustToolVisible()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var onOff = ViewONE.isAdjustToolVisible();

This method returns the visibility of the adjust tool (brightness/contrast/luminance).

Method: **setAppletEnabled(true/false)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.172 or later***
IBM Daeja ViewONE Professional V1.1.172 or later
No additional modules

Example: ViewONE.setAppletEnabled(false);

This method enables / disables the applet GUI.

Method: **setImageToolbarCollapsed(true/false)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.184or later***
IBM Daeja ViewONE Professional V1.1.184or later
No additional modules

Example: ViewONE.setImageToolbarCollapsed(false);

This method puts the image toolbar into an expanded or collapsed (folds) state.

Method: **setViewToolbarCollapsed(true/false)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.184or later***
IBM Daeja ViewONE Professional V1.1.184or later
No additional modules

Example: ViewONE.setViewToolbarCollapsed(false);

This method puts the view toolbar into an expanded or collapsed (folds) state.

Menus and keys

Method: **setFileMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard V3.1.184or later***
IBM Daeja ViewONE Professional V1.1.184or later
No additional modules

Example: ViewONE.setFileMenus(true);

Specifies whether the file pop-up menus are available (using the middle or right mouse button). A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **IsFileMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isFileMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setViewMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setViewMenus(true);

Specifies whether the view pop-up menus are available (using the middle or right mouse button). A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Method: **isViewMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isViewMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setImageMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setImageMenus(true);

Specifies whether the image pop-up menus are available (using the middle or right mouse button). A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Method: **isImageMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isImageMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setPrintMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setPrintMenus(true);

Specifies whether the print pop-up menus are available (using the middle or right mouse button). A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isPrintMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isPrintMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setPageMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setPageMenus(true);

Specifies whether the page pop-up menus are available (using the middle or right mouse button). A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Method: **isPageMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isPageMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setSelectMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setSelectMenus(true);

Specifies whether the select pop-up menus are available (using the middle or right mouse button). A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Method: **isSelectMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isSelectMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setPreferenceMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setPreferenceMenus(true);

Specifies whether the preference pop-up menus are available (using the middle or right mouse button). A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Method: **isPreferenceMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isPreferenceMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setAllMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setAllMenus(true);

Specifies whether the all pop-up menus are available. A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isAllMenus()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var menusEnabled = ViewONE.isAllMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setCacheMenus(true/false)**

Requirements: ***IBM Daeja ViewONE Professional V1.0.808 or later***
No additional modules

Example: ViewONE.setCacheMenus(true);

Specifies whether the cache pop-up menus are available. A value of 'true' (default) indicates the menus are available and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isCacheMenus()**

Requirements: ***IBM Daeja ViewONE Professional V1.0.808 or later***
No additional modules

Example: var menusEnabled = ViewONE.isCacheMenus();

Returns a Boolean value of 'true' if the menus are enabled else a value 'false' is returned.

Method: **setFileKeys(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setFileKeys(true);

Specifies whether the hot keys for file operations are enabled. A value of 'true' (default) indicates the keys are enabled and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isFileKeys()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var keysEnabled = ViewONE.isFileKeys ();

Returns a Boolean value of 'true' if the keys are enabled else a value 'false' is returned.

Method: **setImageKeys(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setImageKeys(true);

Specifies whether the hot keys for image operations are enabled. A value of 'true' (default) indicates the keys are enabled and 'false' indicates they are not.

Method: **isImageKeys()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var keysEnabled = ViewONE.isImageKeys ();

Returns a Boolean value of 'true' if the keys are enabled else a value 'false' is returned.

Method: **setPrintKeys(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setPrintKeys(true);

Specifies whether the hot keys for print operations are enabled. A value of 'true' (default) indicates the keys are enabled and 'false' indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isPrintKeys()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var keysEnabled = ViewONE.isPrintKeys ();

Returns a Boolean value of 'true' if the keys are enabled else a value 'false' is returned.

Method: **setViewKeys(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setViewKeys(true);

Specifies whether the hot keys for view operations are enabled. A value of 'true' (default) indicates the keys are enabled and 'false' indicates they are not.

Method: **isViewKeys()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var keysEnabled = ViewONE.isViewKeys ();

Returns a Boolean value of 'true' if the keys are enabled else a value 'false' is returned.

Method: **setPageKeys(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setPageKeys(true);

Specifies whether the hot keys for page operations are enabled. A value of 'true' (default) indicates the keys are enabled and 'false' indicates they are not.

Method: **isPageKeys()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var keysEnabled = ViewONE.isPageKeys ();

Returns a Boolean value of 'true' if the keys are enabled else a value 'false' is returned.

Method: **setSelectKeys(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setSelectKeys(true);

Specifies whether the hot keys for select operations are enabled. A value of 'true' (default) indicates the keys are enabled and 'false' indicates they are not.

Method: **isSelectKeys()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var keysEnabled = ViewONE.isSelectKeys ();

Returns a Boolean value of 'true' if the keys are enabled else a value 'false' is returned.

Method: **setAllKeys(true/false)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.setAllKeys(true);

Specifies whether the all hot keys are enabled or not. A value of ‘true’ (default) indicates the keys are enabled and ‘false’ indicates they are not.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isAllKeys()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var keysEnabled = ViewONE.isAllKeys ();

Returns a Boolean value of ‘true’ if the keys are enabled else a value ‘false’ is returned.

Opening and saving annotations

Method: **setAnnotationFile(*path*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

IBM Daeja 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 (such as CGI, EXE, or ASP) that streams an annotations file to the applet.

Parameters may be added to the URL, such as user ID or document ID, 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 IBM Daeja ViewONE annotations file specification described in the "Annotations File Format" section of the *IBM Daeja ViewONE Annotations Module Setup Guide*.

If you are unsure whether IBM Daeja ViewONE has processed your annotations file correctly then you can enable the annotations "debug" output (TraceAnnotation).

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

Method: **setAnnotationSavePost(*path*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

Specifies the location of a server-side object (such as CGI, EXE, ASP, or JSP) 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 IBM Daeja ViewONE annotations save POST objects.

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

If you are unsure what IBM Daeja ViewONE is sending through to the object then you can enable the annotations “debug” output (TraceAnnotation parameter).

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 presses the annotations save button.

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

Method: **setAnnotationPostPrefix(*parameters*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

This method specifies the parameters to be added to the POST data for the annotationSavePost feature. 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, and so on.

Method: **setAnnotationSaveServlet(*path*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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 IBM Daeja ViewONE annotations save servlets.

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 – see IBM Daeja ViewONE Parameters Reference Manual for detail), such as user ID or document ID, to send user or document-specific annotations.

If you are unsure what IBM Daeja ViewONE is sending through to the servlet then you can enable the annotations “debug” output.

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 presses the annotations save button.

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

Method: **setAnnotationSave(*path*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

Specifies the location of a server-side object (such as CGI, EXE, ASP, or JSP) 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 IBM Daeja ViewONE annotations save GET objects.

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, to send user or document-specific annotations. IBM Daeja 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 IBM Daeja ViewONE is sending through to the object then you can enable the annotations "debug" output.

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 presses the annotations save button.

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

Method: **saveAnnotations()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.saveAnnotations();

Calling this method will cause IBM Daeja ViewONE to save annotations with whatever server-side annotations save object has been specified (by using the relevant HTML parameters or JavaScript methods).

Important Note: This method is asynchronous so the viewer will not "wait" for the save process to be completed before proceeding so you should ensure that where you use this method and need the operation to be completed before proceeding, you use the event handler and monitor for event 24 ("annotations saved OK") or event 25 ("annotations save failed").

Method: **reloadAnnotations()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
IBM Daeja ViewONE Virtual, any version
Annotations module

Forces IBM Daeja 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".

User interface

Method: **setAnnotationHideButtons(*String*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.setAnnotationHideButtons("note,text,select,arrow");

ViewONE.setAnnotationHideButtons(""); // to clear the list of hidden buttons

This method 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 terms described in the documentation for the configuration parameter "annotationHideButtons" as described in the IBM Daeja ViewONE Parameters Reference Manual To clear the list of hidden buttons, call the method with an empty string.

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

Editing and finding annotations

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

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.setAnnotateEdit(true);

When this method is called with “true”, the user may edit editable annotations and add new annotations by using 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: **setDefaultSelectAnnotation(boolean)**

Requirements: ***IBM Daeja ViewONE Standard V3.0.728 or later***
IBM Daeja ViewONE Professional V1.0.728 or later
Annotations module

ViewONE.setDefaultSelectAnnotation(true);

When this method is called with “true” then the annotation selection button will always be switched ‘on’ (pressed).

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

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.setAnnotateEdit(true);

When this method is called with “true”, the user may edit editable annotations and add new annotations by using 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: **isAnnotationsUpdated()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

var boolAnnotateUpdated = ViewONE.isAnnotationsUpdated();

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

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

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.showAnnotationToolbar(false);

Specifies whether to show or hide the main annotations toolbar.

Method: **isAnnotationToolbar()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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:

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.
An 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*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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() method.

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

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

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

Returns the properties of the annotation on the current page 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: **getAnnotationOnPage(*label,page*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

var strAnnotationProperties = ViewONE.getAnnotationOnPage("myHighlight1",1);

Returns the properties of the annotation that has a label property matching the one specified on the specific page defined by the page parameter. The properties are returned in a delimited list. The convenience method, getDelimiter() can be called to identify the delimiter used.
Note that if there is more than one annotation with the same label name then the result will be undefined.
If there is no match or JavaScript is not ready then the method returns "NONE".

If the "annotationJavaScript" extensions parameter is not enabled then this method will return "OPTION DISABLED"

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

Method: **getActiveAnnotation()**

Requirements: ***IBM Daeja ViewONE Standard V4.0.14 or later***
IBM Daeja ViewONE Professional V4.0.14 or later
Annotations module

var annotationLabel = ViewONE.getActiveAnnotation();

Returns the label of the currently "active" annotation. There is only ever one active annotation that is currently being edited. When there is no active annotation the text "NONE" is returned.

Method: **addAnnotation(annotationProperties)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

Adds an annotation with the specified properties. The *annotationProperties* 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 the *IBM Daeja ViewONE Annotations Module Setup Guide*.

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)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

```
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 the IBM Daeja ViewONE Annotations Module Setup Guide.

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: **startModifyAnnotations()**

Requirements: ***IBM Daeja ViewONE Standard V3.1.100 or later***
IBM Daeja ViewONE Professional V1.1.100 or later
Annotations module

ViewONE.startModifyAnnotations();

If multiple annotations are to be modified together, then it is strongly advised that this method is called before calling modifyAnnotation() method , followed by calling endModifyAnnotations(). By encapsulating multiple calls to modifyAnnotation() with these two methods you will prevent multiple repaints (which are usually generated after every call to modifyAnnotation()). Repainting multiple times can cause flicker and will perform slower than one single repaint!

Examples:

document.viewONE.startModifyAnnotations();

document.viewONE.modifyAnnotation("line1", <annotation properties>);
document.viewONE.modifyAnnotation("line2", <annotation properties>);

document.viewONE.endModifyAnnotations();.

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

Method: **endModifyAnnotations()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.endModifyAnnotations();

See the comments for startModifyAnnotations() for further details

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

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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() method.

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

Method: **getDelimiter()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

var strDelimiter = ViewONE.getDelimiter();

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

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

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.setDelimiter("||");

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

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

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

```
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() method and is really just a convenience method to save parsing in JavaScript.

Method: **startAnnotation(*type*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.startAnnotation("text");

Gets IBM Daeja 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
redact
redactPoly
poly
openPoly
oval
circle
freehand
ruler
angle
anglereversed
stamp
stampMenu<*N*>

Where "stampMenu<*N*>" is used, *N* represents the position of the stamp item in the stamp menu list. The first 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*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

Example: "<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 (user name) 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	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*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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(*TEXTorURL, displayText*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

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

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.

This method can create a menu item for an image stamp or a text stamp.

To specify an image stamp, the *TEXTorURL* parameter must start with the text “image:” and followed by the URL. It can point to any image file that can be decoded by ViewONE.

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.

To specify a text stamp, the *TEXTorURL* parameter is the text that will be displayed in the text stamp annotation.

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

Method: **insertAnnotationStamp(text, beforeIndex, properties)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

var ok = ViewONE.insertAnnotationStamp("my stamp", 0, null);

Calling this method adds a new text stamp to the annotation stamp menu. The beforeIndex parameter specifies the index in the stamp annotations menu that the new stamp should be inserted (starting at 0 for the first element). If an index greater than the number of stamps is specified, the new stamp is added to the end.

The properties parameter is used to specify any properties for the new menu item. For details of the format to use for this parameter see the startAnnotationWithProperties() JavaScript method. Setting this parameter to null will result in default parameters being set for the menu item.

This method returns "true" if the stamp was successfully inserted or "false" otherwise.

Method: **removeAnnotationStamp(*index*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

var ok = ViewONE.removeAnnotationStamp(0);

Calling this method removes the annotation stamp from the annotation stamps menu at the specified index. The menu items in the menu are indexed from 0.

This method returns "true" if the stamp was successfully removed or "false" otherwise.

Method: **setAnnotationStampText(*text, index*)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

var ok = ViewONE.setAnnotationStampText("my stamp", 0);

Calling this method sets the text for an annotation stamp menu item to the specified text. The menu item affected is specified by the index parameter (starting at 0 for the first menu item in the menu).

This method returns "true" if the stamp was successfully changed or "false" otherwise.

Method: **clearAnnotationStamps()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
Annotations module

ViewONE.clearAnnotationStamps();

Calling this method clears the contents of the annotation stamp menu.

Method: **endAnnotation ()**

Requirements: ***IBM Daeja ViewONE Standard V3.1.118 or later***
IBM Daeja ViewONE Professional V1.1.118 or later
Annotations module

ViewONE. endAnnotation();

Calling this method ends the editing of the current annotation and stores any changes internally. It can also be used to end the automatic re-selection of the current annotation type when the current new annotation has been finished editing.

Method: **setRedactionIsSemiTransparent (boolean)**

Requirements: ***IBM Daeja ViewONE Standard V3.0.332 or later***
IBM Daeja ViewONE Professional V1.0.332 or later
DEPRECATED since V4.0.10
Annotations module

ViewONE.setRedactionIsSemiTransparent (true);

Calling this method with the value "true" puts IBM Daeja ViewONE into semi-transparent redaction mode. In this mode redaction annotations become semi-transparent so that the user can see the redacted text and the position of the annotation.

Calling this method with the value "false" puts IBM Daeja ViewONE back into normal redaction mode.

To change the color used for the semi-transparent redaction annotations, use the HTML parameter "transparentRedactionColor "

NOTE: This JavaScript method is DEPRECATED as of Version 4.0.10 replaced by: setAnnotationIsSemiTransaprent(Boolean)

Method: **setAnnotationsSemiTransparent (boolean, type)**

Requirements: ***IBM Daeja ViewONE Standard V4.0.10 or later***
IBM Daeja ViewONE Professional V4.0.10 or later
Annotations module

ViewONE.setAnnotationsSemiTransparent (true, "all");

Calling this method with the value "true" puts IBM Daeja ViewONE into semi-transparent mode. In this mode annotations become semi-transparent so that the user can see the image behind the annotations, so that annotations can be positioned.

The type argument can have one of the following values:

- "all" – all editable annotations are made semi-transparent.
- "redacttypes" – all redact and redactpoly annotation types are made semi-transparent. The method with this parameter replaces the deprecated function setRedactionsIsSemiTransparent().
- "burnable" – all annotations whose type can be permanently redacted. This is set by using the "annotationDefaults" html parameter. See the *IBM Daeja ViewONE Parameters Reference Manual* for details of the "burnable" property.

Calling this method with the value false puts IBM Daeja ViewONE back into normal annotation mode.

To change the color used for the semi-transparent redaction annotations, use the HTML parameter "transparentRedactionColor "

V3 security note: Both methods are disabled by default unless the "annotationJavaScriptExtensions" parameter is set to "true".

Method: **isAnnotationsSemiTransparent (type)**

Requirements: ***IBM Daeja ViewONE Standard V4.0.38 or later***
IBM Daeja ViewONE Professional V4.0.38 or later
Annotations module

var isRedactSemi = ViewONE.isAnnotationsSemiTransparent ("redacttypes");

This method complements the setAnnotationsSemiTransparent method and allows you to query which (if any) annotation types are currently semi-transparent.

The type argument can have one of the following values:

- "all" – will return "true" if all annotations on the document are semi-transparent.
- "redacttypes" – will return "true" where all annotations of the redaction type on the document are semi-transparent.
- "burnable" – will return "true" where all annotations whose type can be permanently redacted are semi-transparent. The "burnable" property is set by using the "annotationDefaults" html parameter. See the *IBM Daeja ViewONE Parameters Reference Manual* for details of the "burnable" property.

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

Method: **setAnnotationHideContextButtons ()**

Requirements: ***IBM Daeja ViewONE Standard V4.0.6 or later***
IBM Daeja ViewONE Professional V4.0.6 or later
Annotations module

ViewONE.setAnnotationHideContextButtons("freehand, hyperlink, text);

This method 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:

	save	Fix button
	security	Edit security button
	text	Text editor button
	change font	Annotation font 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
	strikethrough	Strike through text (IBM Daeja ViewONE Standard V3.0.332 or later, IBM Daeja ViewONE Professional V1.0.332 or later)
	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 text that is used, for example, "strikethrough," is case-insensitive. If all terms are disabled the user will not see the context-sensitive toolbar.

The default is to display all buttons.

Method: **setAnnotationHideContextButtonsIds()**

Requirements: ***IBM Daeja ViewONE Standard V4.0.6 or later***
IBM Daeja ViewONE Professional V4.0.6 or later
Annotations module

ViewONE.setAnnotationHideContextButtonsIds1("note, text);

<PARAM NAME="annotationHideContextButtonsIds1" value="note, text">

This method is used with the setannotationHideContextButtons() method 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 annotation types are case insensitive.

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

Method: **setRulerScale (double)**

Requirements: ***IBM Daeja ViewONE Standard V3.0.794 or later***
IBM Daeja ViewONE Professional V1.0.794 or later
Annotations module

ViewONE. setRulerScale (1.0);

Calling this method with the value sets the scale for all the ruler annotations subsequently added.

Method: **setRulerUnits (text)**

Requirements: ***IBM Daeja ViewONE Standard V3.0.794 or later***
IBM Daeja ViewONE Professional V1.0.794 or later
Annotations module

ViewONE. setRulerScale ("cm");

Calling this method sets the units the ruler annotation displays. The values are:

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

Timeout/User Idle Control

Method: **setTimeout(seconds)**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var ready = ViewONE.setTimeout(30);

This method sets and starts a usage timer. If the user does not use the applet for the number of seconds specified then the applet will automatically be disabled. It can be re-enabled by calling one of the timeout JavaScript methods (see below), opening a document using one of the JavaScript open methods, by revisiting the page containing the applet (Netscape) or reloading the page (Internet Explorer).

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **getTimeout()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var seconds = ViewONE.getTimeout();

This method returns the timeout value (integer seconds) set using either the setTimeout() method or the HTML tag "timeout".

Method: **stopTimeout()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.stopTimeout();

This method will disable the timer set using either the setTimeout() method or the HTML tag "timeout" and if the applet had timed-out then it will wake up (that is, be re-enabled).

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **isTimedOut()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var timedout = ViewONE.isTimedOut();

This method returns a value of true if the applet has timed-out as a result of the user not using the applet for the time specified by the setTimeout() method or the HTML tag "timeout". It otherwise returns false.

Method: **getTimeLeft()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: var timeleft = ViewONE.getTimeLeft();

This method returns the time in seconds left before the applet times-out, but only if the setTimeout() method or HTML tag "timeout" has been used to set the time in the first place. It otherwise returns 0. The time left is automatically reset each time the user performs any action, such as scrolling, changing pages, or zooming.

Method: **wakeUp()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example: ViewONE.wakeUp();

This method will wake up the applet if it has timed out (see setTimeout()).The user will then be able to use the applet as normal.

Security note: This method is disabled by default unless the "JavaScriptExtensions" parameter is set to "true".

Method: **setEventHandlerResponse(text)**

Requirements: ***IBM Daeja ViewONE Standard V3.0.610 or later***
IBM Daeja ViewONE Professional V1.0.610 or later
No additional modules

Example: ViewONE.setEventHandlerResponse("url:http://www.daeja.com/test.tif");

This method is for use with the event handler (see next section). It is used when events 44 and 45 are generated and its purpose is to inform the viewer of alternative URLs for the current document/page. See event handler notes for events 44 and 45 in Appendix A.

Method: **setServerEventHandler()**

Requirements: ***IBM Daeja ViewONE Standard, any version***
IBM Daeja ViewONE Professional, any version
No additional modules

Example:
ViewONE.setServerEvetnHandler("http://www.Mysite.com/MyServerLogging.aspx");

Sets the server event handler URL to the specified URL – see "ServerEventhandler" documentation in IBM Daeja ViewONE Parameters Reference Manual for further details.

Burner Methods

Method: **burnAnnotations(prompt)**

Requirements: ***IBM Daeja ViewONE Professional V1.1.96 or later***
PRS module

This JavaScript method activates the burn process with whatever server-side permanent redaction object that has been defined. It calls the same functionality that pressing the “Burn” button calls.

The **prompt** argument defines whether the user should be prompted to accept or reject the burn before it is activated. The values are “true” and “false”.

Method: **setBurnPDFToPDF(true/false)**

Requirements: ***IBM Daeja ViewONE Professional V1.1.172 or later***
PRS module

This JavaScript method replicates the behavior of the html parameter "burnPDFToPDF". When a burn request is sent from the client to the server-side component, the value set by this JavaScript method indicates if the source document is a PDF document, the burnt output document should also be a PDF document.

See the HTML parameter "burnPDFToPDF" in the *IBM Daeja ViewONE Parameter Reference Manual* for more details.

Streamer Methods

Method: **setStreamerEnabled(true\false)**

Requirements: ***IBM Daeja ViewONE Professional V1.1.184 or later***
Document Streaming Server module

Example: ViewONE.setStreamerEnabled(true);

The setStreamerEnabled() is the JavaScript equivalent of the "StreamerEnabled" HTML parameter. This method allows the streamer to be enabled and disabled dynamically.

Method: **setStreamerURL(url)**

Requirements: ***IBM Daeja ViewONE Professional V1.1.184 or later***
Document Streaming Server module

Example: ViewONE.setStreamerURL(http://myStreamerHost/Streamer);

The setStreamerURL() is the JavaScript equivalent of the "StreamerURL" HTML parameter. This method allows the streamer location to be set dynamically.

Important Note: When opening a document by using JavaScript, the setStreamerURL(String url) call will need to be made before each call to open a document in the same way as the setAnnotationSource(String file) method.

The Event Handler and Event Handling

IBM Daeja ViewONE introduces the concept of applet JavaScript event handling. This is a mechanism by which it is possible to use JavaScript to monitor user activity and other selected actions performed by the IBM Daeja ViewONE applet.

Typically when using the viewers JavaScript API, you should be expecting to make use of the viewers event handler to actively control the flow of events because the viewer is an asynchronous threaded application – so when using the JavaScript methods, it is important to understand that the viewer does not necessarily "wait" for one thread to finish before starting another and you can find you get undesired results when calling one method directly after another.

For example, suppose you want to use the setZoom() method to zoom the document in to a particular zoom level on opening. Now if you just load the viewer and call the "setZoom()" method immediately, it may or may not work depending on how long it takes to render the image. If the zoom method is called BEFORE the document has been fully rendered, it will not work since it requires the rendered image to zoom in on.

So if this is not coded "defensively" with use of the event handler, users can see intermittent "faults" showing up in the code based on the timing of events. So using the zoom example above, for small documents which load quickly, the zoom function works because the image gets rendered before the zoom method is called. But for larger documents, users may see the zoom does not get set because the image has not yet finished rendering when the zoom method is called.

So the correct approach for this example would be to have the event handler monitor for event 38 (viewer has been stated and page rendered) and only after this has been fired would you then call the setZoom() method.

The event handling is enabled by the use of the HTML parameters "eventHandler" and “eventInterest” as described in detail in the IBM Daeja ViewONE Parameters Reference Manual . The "eventHandler" sets the name of the JavaScript function that gets called and the “eventInterest” specifies the event Id’s that are wanted (see Appendix A for full list of Id’s).

When the viewer is initialized, it will call this function, passing its parameters relevant to the action performed as described below.

The JavaScript ‘event handler’ must be specified with two parameters; id (integer) and text (String), but can be any name of your choice, as follows:

```
Function myEventHandler(id, text)
{
  alert("Event received, id="+id+", text="+text);
}
```

You are however free to do whatever you need, for example testing the value of the id and text fields then taking further actions as necessary.

NOTE: The MayScript tag must also be included in the applet definition for the event handler to work correctly.

Event handler change in IBM Daeja ViewONE V4.0.38 or later

Since version 4.0.38 of the viewer, the event handling has been extended to allow additional detail to be passed back with the event in the form of a JSON object which can be parsed to further customize the viewer.

This additional functionality is enabled by setting the HTML parameter "eventHandlerJSON" to "true" (see IBM Daeja ViewONE Parameters Reference Manual for additional detail on this parameter) and the second worked example shows one way it might be used.

With this parameter enabled, the event handler now takes the form:

```
function myEventHandler(id, text,json)
{
   alert("Event received, id="+id+", text="+text+ "JSON="+json);
}
```

Note that only certain events support the JSON object for inclusion of additional detail and as detailed in the list of supported events in Appendix A.

Appendix A: Events ids and descriptions

IBM Daeja ViewONE can signal many different events to the event handler.

The following list describes the events that will be received by this function (Note the "HTML Support" column indicates if the event is available in the HTML version of the viewer):

Id	HTML support	JSON support	Event Text	Description
0				
1				User has rotated a page.
3			printpage: page *n* of *n*	User has printed a page.
4			printvisible: page *n* of *n*	User has printed the visible part of a page
5	√		opened: *url*	User has opened a document
6			saved: page *n* of *n*	User has saved a page
7		√ 4.0.32 or later	click: page *n* of *n*	User has clicked on a page JSON object {"pageX": <value>, "pageY": <value>, "page": {"number": <value>, "view": {"resolutionX": <value>, "resolutionY": <value> }}}
8		√ 4.0.32 or later	dblclick: page *n* of *n*	User has double-clicked on a page JSON object {"pageX": <value>, "pageY": <value>, "page": {"number": <value>, "view": {"resolutionX": <value>, "resolutionY": <value> }}}
9			page: page *n* of *n*	User has changed page
10			timeout: page *n* of *n*	Indicates the applet has timed-out. See JavaScript method setTimeout() and HTML tag "timeout"
11				Reserved.
12			select page: page *n* of *n*	User has selected a page
13			unselect page: page *n* of *n*	User has unselected a page

Id	HTML support	JSON support	Event Text	Description
14		√ 4.0.46 or later	mouse down: page *n* of *n*	User has pressed a mouse button JSON object {"pageX": <value>, "pageY": <value>, "page": {"number": <value>, "view": {"resolutionX": <value>, "resolutionY": <value> }}}
15		√ 4.0.46 or later	mouse up: page *n* of *n*	User has released a mouse button JSON object {"pageX": <value>, "pageY": <value>, "page": {"number": <value>, "view": {"resolutionX": <value>, "resolutionY": <value> }}}
16				Reserved.
17				Reserved.
18				Reserved.
19				Reserved.
20	√		set document: doc *n* of *n*	User has selected the next document in the list (this applies only when the "doc<N>" HTML tag is used – see HTML manual)
21			end tab	The user has used the tab key to change focus while the last focusable component in IBM Daeja ViewONE already had focus. IBM Daeja ViewONE will assign focus back to the first in its list, however you may override IBM Daeja ViewONE by using this event to switch focus to any alternative component on your web page.
22	√		ready	The applet has just been started and is ready to accept JavaScript calls.
23			annotation hyperlink	The user has activated an annotation (JavaScript) hyperlink (see annotations configuration manual).
24	√		annotations save ok	Annotations have just been saved.
25	√		annotations save failed	The annotation save operation has just failed.
26			print canceled	A print dialog or print job has been canceled.
27			print ended	A print job has been successfully sent to the printer.
28				Reserved.
29				Reserved.
30			annotation created	The user has added an annotation through the user interface.

Id	HTML support	JSON support	Event Text	Description
31			SaveDocument Failed(*n*)	This event can fire with two different *n* values: - if *n* is 1, it means that ViewONE could not create the save files in its own cache (before sending them to the destination). One way this might happen is if the disk was full. - if *n* is 2, it means that ViewONE could not copy the save files to the specified destination.
32			annotations updated: page *n* of *n*	Gets fired the first time a document's annotations set is modified. It will not get fired again until such time as the annotation data has been saved and then modified again following the save. It can, therefore, be used to indicate that the document's annotations now require saving.
33			annotations restored: page *n* of *n*	Gets fired whenever a document's annotations set is restored. It can, therefore, be used to indicate that the document's annotations no longer require saving.
34			Page rendered	The image for the current page has been rendered / updated. This event occurs on every redraw/refresh. See event 38/39 for filtered versions.
35			Area Selected	An area of the current page has been selected by the user (using the zoom area tool).
36				Reserved

<table>
<tr><th>Id</th><th>HTML support</th><th>JSON support</th><th>Event Text</th><th>Description</th></tr>
<tr><td>37</td><td></td><td></td><td>Key pressed</td><td>
A key has been pressed (key defined in the event).

Notes:

It is strongly recommended that you turn off key processing by IBM Daeja ViewONE by using the HTML parameter "ProcessKeys" set to False, otherwise IBM Daeja ViewONE will process keys AND generate events.

It is also strongly recommended that you set up a test to display the event text before assuming that value of that text because some keyboards may differ in the text and may also differ according to the localization setup of your machine.

The following keys do not generate events..

- Keys processed when editing a text annotation
- SHIFT, CTRL and ALT Keys (instead, when a key is pressed a marker is included to indicate when these keys are pressed with other keys –see examples below)
- Windows main menu key
- Popup menu key

The following keys will generate events but will always also be handled by IBM Daeja ViewONE event when ProcessKeys is set to False:

- Scroll bar keys (Page up/down, home, end, arrow keys)
- Windows menu key

Modifier and cursor keys are defined using names (see below)

Examples:

<table>
<tr><th>Event Text</th><th>Description</th></tr>
<tr><td>"A"</td><td>A key pressed</td></tr>
<tr><td>Control A</td><td>Ctrl + A keys pressed</td></tr>
<tr><td>Shift A</td><td>Shift + A keys pressed</td></tr>
<tr><td>Alt A</td><td>Alt + A keys pressed</td></tr>
<tr><td>Alt Shift Control A</td><td>Alt Shift + Ctrl + A keys pressed</td></tr>
<tr><td>Space</td><td>Space key pressed</td></tr>
<tr><td>Ctrl Space</td><td>Ctrl + Space keys pressed</td></tr>
<tr><td>Caps Lock</td><td>Caps Lock pressed</td></tr>
<tr><td>Up</td><td>Up arrow key pressed</td></tr>
<tr><td>Numpad-9</td><td>9 key on a number pad is pressed (note some numpad keys have a ‘-‘ separator rather than a space)</td></tr>
</table>
</td></tr>
</table>

Id	HTML support	JSON support	Event Text	Description
38			Full page rendered	A page has been viewed in the full-page panel. This event only occurs the first time the page is viewed.
39			All full pages rendered	All pages in the document have been viewed in the full-page panel. This event occurs only once per document.
40				Reserved
41			Keep Alive	This event is generated at regular intervals (specified by the "KeepAliveTime" HTML parameter). The text accompanying the event is specified by the "KeepAlive" HTML parameter.
42			Save Complete	Generated when a document is successfully saved.
43			Cache Access Denied	This event is generated if all attempts to write to an IBM Daeja ViewONE cache have failed.
44			Request alternative URL	When this event is enabled, then it is generated each time and just before the viewer retrieving a document/image. The event supplies the URL which is about to be used. The Event Handler may return a different URL (using the setEventHandlerResponse() method, passing an alternative URL using the format "url:<url>") in which case the viewer will use the alternative supplied URL.
45			Request alternative URL upon error	When this event is enabled, then it is generated when the viewer encounters a FileNotFound condition while retrieving a document/image. The event supplies the URL which was used. The Event Handler may return a different URL (using the setEventHandlerResponse() method, passing an alternative URL using the format "url:<url>") in which case the viewer will again try using alternative supplied URL.
46			Magnifier opened/closed	When this event is enabled, then it is generated when the magnifier is opened or closed.
47			Save dialog cancelled	This event is generated when the user cancels the Save dialog (save document or page)
48		√ 4.0.38 or later	Annotations burnt	This event is generated when annotation burning on the server has finished and the client is notified. JSON object "customServerResponse" Note: you will need to modify your server code to populate the text
49				Reserved.
50				Reserved.
51			Annotation burn canceled by user (Version 3.1.106 onwards)	This event is generated when the burn has been canceled by the user when prompted.

Id	HTML support	JSON support	Event Text	Description
52		√ 4.0.38 or later	Annotation burn failed (Version 3.1.106 onwards)	This event is generated when the client is notified that the annotation burning has failed. JSON object "customServerResponse" Note: you will need to modify your server code to populate the text
57			Print start (version 3.1.158 or later)	This event is generated the moment before a printing operation is started. There is only one event fired per printing operation. For example, printing 3 pages will result in a single event being fired. If a user clicks the print button but cancels the print from the print dialog before printing has started, this event will not occur.
58			Start page (version 3.1.172 or later)	This event is generated when a page is first selected for display. This is before the page gets downloaded (if it needs to be).
59	√		Annotations loaded OK (version 3.1.198 or later)	This event is generated when Annotations have been successfully loaded into the viewer with no errors. Note that if you are streaming annotation data, the event gets generated on a per page basis and the page number is supplied along with the event itself
60	√		Annotations load failed (version 3.1.198 or later)	This event is generated when retrieval of annotation data from the server-side component has failed
61	√		Document load failed (version 4.0.4 or later)	This event is generated when a document has failed to load. This can occur if the document to be displayed cannot be found, or the annotations failed to load or there was a problem with the document so it could not be displayed.
62	√		Page load failed (version 4.0.4 or later)	This event is generated when a document is composed of multiple individual pages and one of those pages could not be displayed. This can occur if the page to be displayed cannot be found or there was a problem with the page so it could not be displayed.
63	√		Thumbnail load failed (version 4.0.4 or later)	This event is generated when the image being displayed in a thumbnail has been generated from separate thumbnails and the thumbnail could not be displayed. This can occur if the thumbnail to be displayed cannot be found or there was a problem with the thumbnail so it could not be displayed.
65	√		Annotation Activated (version 4.0.38 or later)	This event is fired every time an annotation is "activated" – an annotation is "activated" when the user selects it by clicking on it using the mouse. Note that the creation of a text annotation will also cause this event to be fired since the annotation becomes activated (to allow for text entry) as soon the user has finished drawing the outline of the text box.
66	√		Annotation Deactivated (version 4.0.38 or later)	This event is fired every time an annotation is "deactivated" – an annotation is "deactivated" when the user clears it by clicking anywhere on the document away from the currently selected annotation with a left-click of the mouse.

Id	HTML support	JSON support	Event Text	Description
67	√		Annotation Deleted (version 4.0.38 or later)	This event is fired every time the user deletes an annotation using the user interface or JavaScript methodology and returns the label of the deleted annotation. Where multiple annotations are deleted, this method returns an array containing all labels of the deleted annotations. Note that the event does NOT get fired if you remove annotations using the "undo" button.
68		√	Mouse move (JSON support 4.0.42 or later)	User has moved mouse on a page JSON object {"pageX": <value>, "pageY": <value>, "page": {"number": <value>, "view": {"resolutionX": <value>, "resolutionY": <value> }}}

Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/us/en/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, and service names may be trademarks or service marks of others.

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use cookies that collect each user’s user name for purposes of session management, authentication, and enhanced user usability. These cookies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Product Number: 5725-Q02, 5725-Q03, 5725-Q04

SC27-6387-00