VirtualViewer®

V3.3 VirtualViewer HTML5 API User's Guide

Note:
An online version of this manual contains information on the latest updates to VirtualViewer. To find the most recent version of this manual, please visit the online version at www.virtualviewer.com or download the most recent version from our website at www.snowbound.com/support/manuals.html.

DOC-0168-01

Copyright Information

While Snowbound® Software believes the information included in this publication is correct as of the publication date, information in this document is subject to change without notice.

UNLESS EXPRESSLY SET FORTH IN A WRITTEN AGREEMENT SIGNED BY AN AUTHORIZED REPRESENTATIVE OF SNOWBOUND SOFTWARE CORPORATION MAKES NO WARRANTY OR REPRESENTATION OF ANY KIND WITH RESPECT TO THE INFORMATION CONTAINED HEREIN, INCLUDING WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PURPOSE. Snowbound Software Corporation assumes no responsibility or obligation of any kind for any errors contained herein or in connection with the furnishing, performance, or use of this document.

Software described in Snowbound documents (a) is the property of Snowbound Software Corporation or the third party, (b) is furnished only under license, and (c) may be copied or used only as expressly permitted under the terms of the license.

All contents of this manual are copyrighted by Snowbound Software Corporation. The information contained herein is the exclusive property of Snowbound Software Corporation and shall not be copied, transferred, photocopied, translated on paper, film, electronic media, or computer-readable form, or otherwise reproduced in any way, without the express written permission of Snowbound Software Corporation.

Microsoft, MS, MS-DOS, Windows, Windows NT, and SQL Server are either trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Adobe, the Adobe logo, Acrobat, and the Acrobat logo are trademarks of Adobe Systems Incorporated.

Sun, Sun Microsystems, the Sun Logo, and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

iText, the Initial Developers of the Original Code are Bruno Lowagie and Paolo Soares. Portions created by Bruno Lowagie are Copyright ©1999-2010 by Bruno Lowagie.

Kakadu JPEG2000©, is copyrighted by Dr. David Taubman, and is proprietary to NewSouth Innovations, Pty. Ltd, Australia.

I-NET JWebEngine© is proprietary to I-NET Software GmbH, and shall remain the sole and exclusive property of I-Net Software GmbH.

United States Government Restricted Rights
The Software is provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the United States Government is subject to restrictions as set forth under subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause of DFARS 252.227 –19 or subparagraphs (c)(i) and (2) of the Commercial Computer Software-Restricted Rights at 48 CFR 52.227 – 19 as applicable. The Manufacturer is Snowbound Software Corporation, 309 Waverley Oaks Rd., Suite 401, Waltham, MA 02452, USA.

All other trademarks and registered trademarks are the property of their respective holders.

Manual Title: Snowbound Software VirtualViewer HTML5 Administrator’s Guide
Part Number: DOC-0168-01
Revision: 02
VirtualViewer HTML5 Release Number: 3.3
VirtualViewer® Release Number: 3.3

Printing Date: September 2013

Published by Snowbound Software Corporation.
309 Waverley Oaks Road
Suite 401
Waltham, MA 02452 USA
phone: 617-607-2000
fax: 617-607-2002

©1996 - 2013 by Snowbound Software Corporation. All rights reserved.

Table of Contents

Chapter 1 - Customizing VirtualViewer HTML5 through JavaScript API Methods

This chapter shows how to configure the VirtualViewer HTML5 JavaScript API methods on your system.

You can customize the user interface of VirtualViewer HTML5 by editing JavaScript API methods and customizing the code in ajaxClient.html.

Warning:
Please make a back-up copy of the ajaxClient.html file before you edit it.

You can have a different ajaxClient.html for each type of user, or have a script generate the HTML on the fly.

The ajaxClient.html file contains code that can be customized starting after the following line:

```
<body onload="myFlexSnap.initViaURL()">
```

Please see the example below:

Example 1.1: Customizing What is Displayed in VirtualViewer HTML5

```
<!--
<div id="flipX"
onclick="javascript:myFlexSnap.flipX()"
title="Flip Horizontal"
class="mouseDown"
alt="Flip Horizontal">&nbsp;</div>
-->
```

You can do this with other buttons and menus as well. The descriptions of the options are in JavaScript API.

Document Methods for Setting, Printing, Exporting, and Saving

This section describes the VirtualViewer HTML5 document methods for setting, printing, exporting, and saving.

getDocumentId()

This method returns the `documentId` parameter. The `documentId` is used to identify the document in the active tab of the VirtualViewer HTML5. For example, you could update the status bar for the window with the current `documenId`:

```
window.status = myFlexSnap.getDocumentId();
```

The syntax for the `documenId` is determined by the content handler (also known as a Connector) that is being used by VirtualViewer HTML5. The default content handler is the File Content Handler, so the id is a file name. If using the URL content handler, the id is a URL.

Parameter

The `getDocumentId()` method contains the following parameter:

Parameter	Type	Description
`documentId`	`String`	The name or ID of the document.

getClientInstanceId()

This method returns the `clientInstanceId` parameter. The `clientInstanceId` is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs. This string may be encrypted. Your content handler should implement the encryption and decryption. VirtualViewer HTML5 does not look at this value at all.

Parameter

The `getClientInstanceId()` method contains the following parameter:

Parameter	Type	Description
`clientInstanceId`	`String`	The name or ID of the client instance information. It is often used to hold session or other client specific information that the content handler (Connector) needs

Returns

The `clientInstanceId` of the current document

setClientInstanceId(id)

This method sets the `ClientInstanceId`. The `ClientInstanceId` is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs. This string may be encrypted. Your content handler should implement the encryption and decryption. VirtualViewer HTML5 does not look at this value at all.

Parameter

The `setClientInstanceId(id)` method contains the following parameter:

Parameter	Type	Description
`id`	`String`	The `ClientInstanceId` is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs.

Returns

Undefined

setDocumentId(id)

This method sets the current document id.

Parameter

The `setDocumentId(id)` method contains the following parameter:

Parameter	Type	Description
`id`	`String`	The document id.

Returns

Undefined

setDocumentIdGenerator(fn)

This method sets a callback function to be called when creating a new document.

Parameter

The `setDocumentIdGenerator(fn)` method contains the following parameter:

Parameter	Type	Description
`fn`	`Function`	The function to call when needing a document id.

Returns

Undefined

setExportDocumentNameGenerator(fn)

This method allows a document name to be passed in when the export function is called.

Parameter

The `setExportDocumentNameGenerator(fn)` method contains the following parameter:

Parameter	Type	Description
`fn`	`Function`	The function to call when needing a document id.

Returns

Undefined

setDocumentId(id)

This method sets the current document id.

Parameter

The `setDocumentId(id)` method contains the following parameter:

Parameter	Type	Description
`id`	`String`	The document id.

Returns

Undefined

exportDocument()

This method displays the Export Document dialog box to export any page manipulations or annotation changes to the server. The export downloads the currently active document to the client's local machine. The client is given the choice to save as TIF, PDF, or the original format. If saving in the original format there is no option to include annotations. Please see the example below:

```
onclick="javascript:myFlexSnap.exportDocument()"
```

If `exportBurnAnnotations` in config.js is true and the document includes annotations, then the annotations will be burned into the document. The separate .ann files are not downloaded to the client, so it is not an option to download documents with annotations in their original format.

Returns

Undefined

getPageCount()

The method returns the total number of pages in the currently active document. A negative number indicates an error.

Type

Integer

Returns

The current page count

printDocument()

This method initializes and shows the Print dialog box to print the current document with or without annotations. Please see the example below:

```
onclick="javascript:myFlexSnap.printDocument()"
```

Only visible layers with a Print permission level or higher in the Image Panel will print.

Returns

Undefined

saveDocument(sync)

This method saves the current document including any image manipulations and annotations. Please see the example below:

```
onclick="javascript:myFlexSnap.saveDocument()"
```

The `vvStatusSavingDocument` config.js parameter can be used to display a status message while the document is being saved.

If the `clearCacheOnSave` parameter to true in your web.xml file and `documentCacheSize` is greater than 0, then the old version of the document will be removed from the server's document cache. The `annotationOutputFormat` in web.xml can be used to determine the annotation format to use when saving.

Parameter

The `sendDocument(sync)` method contains the following parameter:

Parameter	Type	Description
`sync`	`boolean`	Whether to make the request asynchronously or not. Due to legacy browser considerations this should be set to true.

Returns

Undefined

sendDocument()

This method sends the document via the content handler by way of the server. Behavior may vary depending on the Connector being used. The default Connector behavior is to create a copy of the document on the server named `"send_<filename.ext>"`.

Some Connectors may use the `emailFromAddress`, `emailSMTPServer` and other configuration settings to send email. The variable `sendDocumentWithAnnotations` in config.js determines whether or not annotations are burned into the copied document. Please see the example below:

```
onclick="javascript:myFlexSnap.sendDocument()"
```

Returns

Undefined

Interacting with Document Pages within the Viewer

This section describes how to configure document pages in the viewer.

closeTab(tabNumber)

This method closes the tab corresponding to `tabNumber`. It removes the tab from the UI and switches the view to a different tab in its place.

Parameter

The `closeTab(tabNumber)` method contains the following parameter:

Parameter	Type	Description
`tabNumber`	`integer`	Zero-based index of the tab to close.

Returns

Undefined

copySelection()

This method copies the currently selected (in thumbnail panel) pages to the *clipboard* (in this context, this is not referring to the system clipboard).

Type

boolean

Returns

True if pages were selected. False if pages were not selected.

cutSelection(delPages)

This method cuts the currently selected (in thumbnail panel) pages to the *clipboard* (in this context, this is not referring to the system clipboard). If `delPages` is true, the pages are simply deleted and not placed on the clipboard.

Parameter

The `cutSelection(delPages)` method contains the following parameter:

Parameter	Type	Description
`delPages`	`boolean`	If true, the pages will be deleted and the clipboard will remain unmodified.

Type

boolean

Returns

True if pages were selected. False if pages were not selected.

despeckleImage()

This method despeckles the image.

Type

boolean

Returns

True if pages are despeckled. False if pages are note despeckled.

getActiveTab()

This method returns the index of the currently selected tab.

Type

Integer

Returns

The zero-based index of the active tab

getBrightness()

This method gets the document's brightness to a particular value between -125 and 125.

Type

Integer

Returns

A value between -125 and 125.

getContrast()

This method gets the document's contrast to a particular value between -125 and 125.

Type

Integer

Returns

A value between -125 and 125.

getGamma()

This method gets the document's gamma to a particular value between -125 and 125.

Type

Integer

Returns

A value between -125 and 125.

getPageNumber()

This method returns the current page number of the page currently being viewed. This method is zero-based. If no page is set, the default value is 1.

Type

Integer

Returns

Zero-based page index

firstPage()

This method switches to the first page.

Returns

Undefined

nextPage()

This method switches to the next page.

Returns

Undefined

pasteSelection(beforePageNum, newDocument)

This method pastes the pages contained on the clipboard into the document.

Parameter

The `pasteSelection(beforePageNum, newDocument)` method contains the following parameters:

Parameter	Type	Description
`beforePageNum`	`Integer`	A zero=based page index specifying where to insert the new pages.
`newDocument`	`boolean`	Used internally when creating a new document from the pages of an existing document.

Returns

Undefined

previousPage()

This method switches to the previous page.

Returns

Undefined

lastPage()

This method switches to the last page.

Returns

Undefined

flipX()

This method rotates the page horizontally along the X axis.

Returns

Undefined

flipY()

This method rotates the page vertically along the Y axis.

Returns

Undefined

invertImage()

This method inverts the colors of the current page.

Returns

Undefined

openInTab(id, newDocument)

This method creates a tab for document with id as `id`. It handles the initialization of a new document within a new tab element.

Parameter

The `openInTab(id, newDocument)` method contains the following parameter:

Parameter	Type	Description
`id`	String	The `documentId` of the document to open
newDocument	boolean	Whether this is a newly created document.

Returns

Undefined

rotateClock()

This method rotates the current document clockwise 90 degrees.

Returns

Undefined

rotateCounter()

This method rotates the current document counter-clockwise 90 degrees.

Returns

Undefined

setBrightness(value)

This method sets the document's brightness to a particular value between -125 and 125..

Returns

Undefined

setContrast(value)

This method sets the document's contrast to a particular value between -125 and 125.

Returns

Undefined

setGamma(value)

This method sets the document's gamma to a particular value between -125 and 125.

Returns

Undefined

showAboutDialog()

This method displays the About dialog box.

Returns

Undefined

toggleLayerManager()

This method toggles the display of the layer manager.

Returns

Undefined

toggleThumbnailPanel(show)

This method toggles the display of the thumbnail panel.

Parameter

The `toggleThumbnailPanel(show)` method contains the following parameter:

Parameter	Type	Argument	Description
`show`	boolean	<optional>	If true, show the thumbnail panel. If false, hide the thumbnail panel. If undefined, toggle the thumbnail panel.

Returns

Undefined

Adjusting the Size of the Window

Note:
The `defaultZoomMode`, `fitLastBetweenDocuments`, `maxZoomPercent`, `zoomTimeout`, and `retainViewOptionsBetweenPages` configuration parameters may affect the behavior of the methods listed below.

fitWidth()

This method zooms the current page to fit its width to the exact width of the viewing area. It display the image at 100% and fills the entire image panel.

Returns

Undefined

fitHeight()

This method zooms the current page to fit its height to the exact height of the viewing area.

Returns

Undefined

fitWindow()

This method zooms the current page to fit the entire page in the viewing area.

Returns

Undefined

getZoomPercent()

This method returns the ratio of image's current height on the screen over its original height. Unfortunately, this method does not actually return a percentage. To obtain the percentage, multiply by 100.

Type

Float

Returns

The zoom ratio

zoomIn()

This method zooms in to the next level on the current document. The first zoom will fit the document to the window, which may be a large increment. Smaller increments occur after the first zoom. The `maxZoomPercent` configuration parameter determines how far the page can be zoomed in.

Returns

Undefined

zoomOut()

This method zooms out to the next level on the current document.

Returns

Undefined

zoomRubberband()

This method activates zoom rubberband mode, allowing the user to specify a rectangle to zoom into using the mouse on the currently selected page. When the user clicks, the display will zoom in to display only the selected section. Please see the example below:

```
onclick="javascript:myFlexSnap.zoomRubberband()"
```

Returns

Undefined

Searching

isDocumentSearchable()

This method determines if the current document is searchable.

Type

boolean

Returns

True if the document is searchable. False if the document is not searchable.

Appendix A - JavaScript API

This appendix lists and describes the JavaScript API for the product.

JavaScript API

Table A.1: Supported JavaScript API Descriptions

Name	Returns	Description
`closeTab(tabNumber)`	Integer	Closes tab corresponding to `tabNumber`. Removes the tab from the UI. Switches view to a different tab in its place. Will return an error if this is the last tab.
`copySelection()`	Boolean	Copies the currently selected (in thumbnail panel) pages to the clipboard (in this context, this is not referring to the system clipboard). Returns true if there were pages selected. Returns false if there were no pages selected.
`cutSelection(delPages)`	Boolean	Cuts the currently selected (in thumbnail panel) pages to the clipboard (in this context, this is not referring to the system clipboard). If delPages is true, the pages are simply deleted and not placed on the clipboard. Returns true if there were pages selected. Returns false if there were no pages selected.
`despeckleImage()`	Boolean	Despeckles image.
`exportDocument()`	Undefined	Displays the Export Document dialog box to export any page manipulations or annotation changes to the server. The export downloads the currently active document to the client's local machine The client is given the choice to save as TIF, PDF, or the original format. If saving in the original format there is no option to include annotations.
`firstPage()`	Undefined	Switches to the first page.
`fitHeight()`	Undefined	Zooms the current page to fit its height to the exact height of the

Name	Returns	Description
		viewing area.
`fitWidth()`	Undefined	Zooms the current page to fit its width to the exact width of the viewing area. It display the image at 100% and fills the entire image panel.
`fitWindow()`	Undefined	Zooms the current page to fit the entire page in the viewing area.
`flipX()`	Undefined	Rotates the page horizontally along the X axis.
`flipY()`	Undefined	Rotates the page vertically along the Y axis.
`getActiveTab()`	Integer	Returns the index of the currently selected tab.
`getBrightness()`	Integer	Gets the document's brightness to a particular value. Between -125 and 125.
`getClientInstanceId()`	String	Returns the `clientInstanceId` parameter. The `ClientInstanceId` is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs. This string may be encrypted. Your content handler should implement the encryption and decryption. VirtualViewer HTML5 does not look at this value at all.
`getContrast()`	Integer	Gets the document's contrast to a particular value. Between -125 and 125.
`getDocumentId()`	String	Returns the `documentId` parameter. The `documentId` is used to identify the document in the active tab of the VirtualViewer HTML5.
`getGamma()`	Integer	Gets the document's gamma to a particular value. Between -125 and 125.
`getPageCount()`	Integer	Returns the number of pages in the currently active document. A negative number indicates an error.
`getPageNumber()`	Integer	Returns the current page number of the page currently being viewed.

Name	Returns	Description
`getZoomPercent()`	Float	Returns the ratio of image's current height on the screen over its original height. Unfortunately, this method does not actually return a percentage. To obtain the percentage, multiply by 100.
`invertImage()`	Undefined	Inverts the colors of the current page.
`isDocumentSearchable()`	Boolean	Returns true if the document is searchable. Returns false if the document is not searchable.
`lastPage()`	Undefined	Switches to the last page.
`nextPage()`	Undefined	Switches to the next page.
`openInTab(id, newDocument)`	Undefined	Creates a tab for document with id as `id`. Handles the initialization of a new document within a new tab element.
`pageContainsAnnotations()`	Undefined	Called during page selection to return a value of true or false indicating if that page has annotations associated with it.
`pasteSelection(beforePageNum, newDocument)`	Undefined	Pastes the pages contained on the clipboard into the document.
`previousPage()`	Undefined	Switches to the previous page.
`printDocument()`	Undefined	Initializes and shows the Print dialog box to print the current document with or without annotations
`rotateClock()`	Undefined	Rotate the current document clockwise 90 degrees.
`rotateCounter()`	Undefined	Rotate the current document counter-clockwise 90 degrees.
`saveDocument(sync)`	Boolean	Saves the current document including any image manipulations and annotations.
`sendDocument()`	Undefined	Sends the document via the content handler by way of the server. The variable `sendDocumentWithAnnotations` in config.js determines whether or not annotations are burned in.
`setBrightness(value)`	Undefined	Sets the document's brightness to a particular value. Between -125 and 125.
`setClientInstanceId()`	String	Allows the `clientInstanceId` to

Name	Returns	Description
		be set via JavaScript method. The `ClientInstanceId` is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs. This string may be encrypted. Your content handler should implement the encryption and decryption. VirtualViewer HTML5 does not look at this value at all.
`setContrast(value)`	Undefined	Sets the document's contrast to a particular value. Between -125 and 125.
`setDocumentId(id)`	String	Sets the current document id.
`setDocumentIdGenerator(fn)`	Undefined	Sets a callback function to be called when creating a new document. Instead of prompting the user for a document id the passed function will be called.
`setExportDocumentNameGenerator(fn)`	Undefined	Allows a document name to be passed in when the export function is called. Pass a function to this method. That function will be called whenever the user clicks Export. The return value of that function will be sent to the servlet and used as the file name of the exported document.
`setGamma(value)`	Undefined	Sets the document's gamma to a particular value. Between -125 and 125.
`showAboutDialog()`	Undefined	Displays the About dialog box.
`toggleLayerManager()`	Undefined	Toggles the visibility of the Layer Manager UI.
`toggleThumbnailPanel(show)`	Undefined	Toggles the display of the thumbnail panel. If true, show the thumbnail panel. If false, hide the thumbnail panel. If undefined, toggle the thumbnail panel.
`zoomIn()`	Undefined	Zooms in to the next level on the current document. The first zoom will fit the document to the window, which may be a large increment.

Name	Returns	Description
		Smaller increments occur after the first zoom. The `maxZoomPercent` configuration parameter determines how far the page can be zoomed in.
`zoomOut()`	Undefined	Zooms out to the next level on the current document.
`zoomRubberband()`	Undefined	Activates the zoom rubber band mode, allowing the user to specify a rectangle to zoom into using the mouse on the currently selected page. When the user clicks, the display will zoom in to display only the selected section.

Index