VirtualViewer®

V3.3 VirtualViewer HTML5 for Java Client Administrator'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.

Snowbound software

DOC-0168-01

Copyright Information

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

Table of Contents

Chapter 1 - Getting Started

Overview

Snowbound Software's VirtualViewer HTML5 viewer works with the latest Java and AJAX technology to create a true zero footprint viewing solution. This chapter will aid you with setting up and working with the package included in your zip file, **VirtualViewerHTML5.zip**. This zip file installs all of VirtualViewer HTML5 components. For information on configuring VirtualViewer HTML5, please see Chapter 3, Customizing the Configuration.

System Requirements

This section describes the system requirements to run VirtualViewer HTML5.

Content Server

VirtualViewer HTML5 requires the VirtualViewer Java Content Server in order to function. The VirtualViewer Java Content Server is included in the VirtualViewerHTML5.zip package.

Servlet Container

VirtualViewer HTML5 requires a J2SE or J2EE servlet container to run. You may choose any compliant servlet container, although recommended servlet containers include Apache Tomcat 4.x and higher, IBM Websphere 5.1 and higher, and BEA Weblogic 8.1 and higher.

Server Java Version

VirtualViewer HTML5 requires a JRE of 1.4.2_10 or higher and we recommend JRE 1.5.

Client Browser Versions

The supported browsers are Internet Explorer 7, 8 and 9, Firefox 2 through 8, or Safari 2 through 5. It may also work with other browsers such as Opera, but no testing is done to ensure compatibility.

Exceptions to Supported File Formats and Platforms

We do our best to support product and document specifications and to work in common platform environments. However, there are always exceptions. If you find an exception, please contact Snowbound Support at http://support.snowbound.com to let us know about it.

Packaging

VirtualViewer HTML5 is delivered as a .zip file including the **VirtualViewerHTML5.zip** installation package. The package may vary depending on your version.

The most current set of documentation is included with the installation package to assist you in installing and administrating this product. Our online documentation available at www.virtualviewer.com is easy to search and has the latest information. The documentation is described below and can be found within the .zip file.

- **VirtualViewerHTML5JavaClientAdminGuide.pdf:** This guide describes how to use and configure VirtualViewer HTML5 Client.
- **VirtualViewerHTML5Javareleasenotes.pdf:** The release notes describe the latest additions and improvements to VirtualViewer HTML5.

What to Expect in an Evaluation Version of VirtualViewer HTML5

Your evaluation is a full version of the product with the following limitations:

- You will see a pop up banner when you view or convert your first document. Subsequent documents in the same session will not elicit the banner.
- You will see large thin Xs across each page after the first 50 pages or thumbnails.
- After your expiration date you will see a banner stating the evaluation has expired. You will not see any output.

Other than that, you will have full use of the product including support for all supported document formats.

What to Expect in a Production Version of VirtualViewer HTML5

When you purchase VirtualViewer HTML5, you will receive a set of fully licensed binary files. The licensed files will be included in VirtualViewerHTML5.war. Please back up your configuration files so that they can be merged into the production version There may be some additional files depending upon the options you purchased.

Installing the Production Version of VirtualViewer HTML5

Install and configure the evaluation version of the product on your target production system. Ensure it is working as you intended.

Extract the binary files from the production version package and use those to replace the same files in the evaluation version that you have installed.

Once the production files are in place, you will no longer see banners or Xs. You will only see expiration messages if you try to view a document of a type that you did not purchase. For example: Office or AFP/MO:DCA.

The evaluation configuration places VirtualViewer HTML5 and the Content Server into the same directory on the same machine. If your environment requires the two servers to be in different directories or on different machines then please contact at http://support.snowbound.com.

Installing

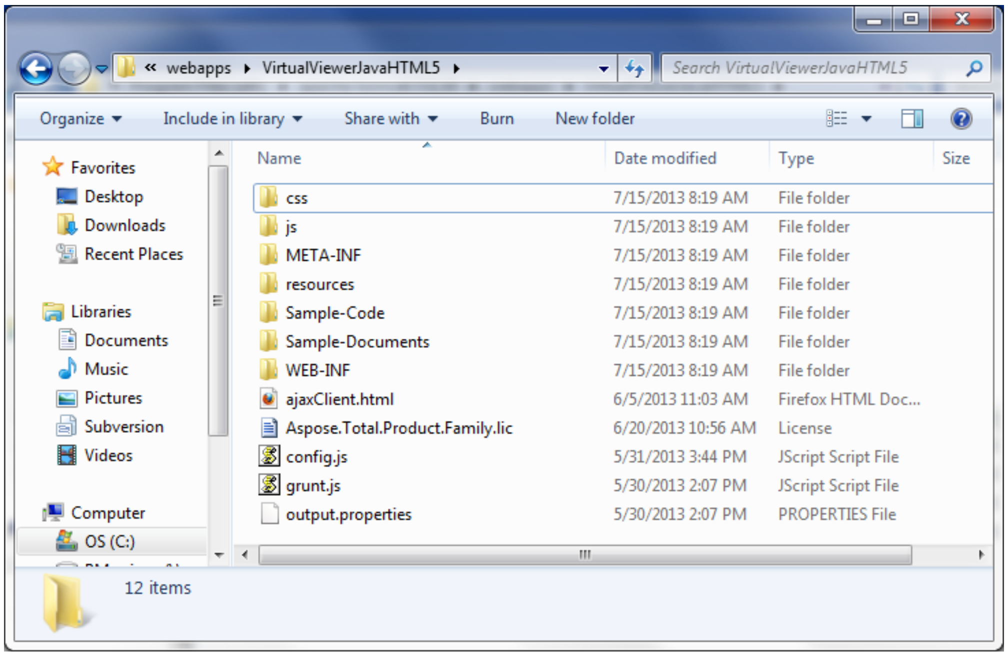
To install VirtualViewer HTML5, follow the steps below:

1. Extract the **VirtualViewerHTML5.zip** file to a directory.

2. The extracted .zip file includes the **VirtualViewerHTML5.war** file.

3. Save the **VirtualViewerHTML5.war** file to the location where you want to install it. Please note that the application needs to be added to a web server before it can be run. Open the **VirtualViewerHTML5.war** file in an archive utility such as 7-zip.

4. In the **VirtualViewerHTML5** directory, you will see the extracted files for VirtualViewer HTML5



Name	Date modified	Type	Size
css	7/15/2013 8:19 AM	File folder	
js	7/15/2013 8:19 AM	File folder	
META-INF	7/15/2013 8:19 AM	File folder	
resources	7/15/2013 8:19 AM	File folder	
Sample-Code	7/15/2013 8:19 AM	File folder	
Sample-Documents	7/15/2013 8:19 AM	File folder	
WEB-INF	7/15/2013 8:19 AM	File folder	
ajaxClient.html	6/5/2013 11:03 AM	Firefox HTML Doc...	
Aspose.Total.Product.Family.lic	6/20/2013 10:56 AM	License	
config.js	5/31/2013 3:44 PM	JScript Script File	
grunt.js	5/30/2013 2:07 PM	JScript Script File	
output.properties	5/30/2013 2:07 PM	PROPERTIES File	

5. Find the web application (webapps) directory where you want to install the files. For example:
`C:\Program Files\Apache Software Foundation\Tomcat 7.0\webapps`

6. From the extracted zip directory, copy the **VirtualViewerHTML5** directory as a new subdirectory under your `webapps` directory. If you are using a web server other than Tomcat this may be a different location.

7. Add the **ServletContext** to the web server by adding the following entry to the **TOMCAT_HOME/conf/server.xml** file:

```
<Context path="/VirtualViewerHTML5" docBase="VirtualViewerHTML5"
debug="0" reloadable="true" />
```

Note:
For other web servers such as WebLogic or WebSphere, you may need to take web application deployment steps that are specific to that type of web server.

8. Verify that the VirtualViewer HTML5 content server is running by looking at the web server logs and search for **VirtualViewer**. You should see the start up message. If you do not see **VirtualViewer**, then search for **snow** which may be in an error message. If you still do not get a result, then please refer to the web application deployment documentation for your web server for information on troubleshooting web application deployment and start up issues.

Note:
We suggest that you restart Tomcat or your web application to ensure that your changes have taken effect.

Verifying

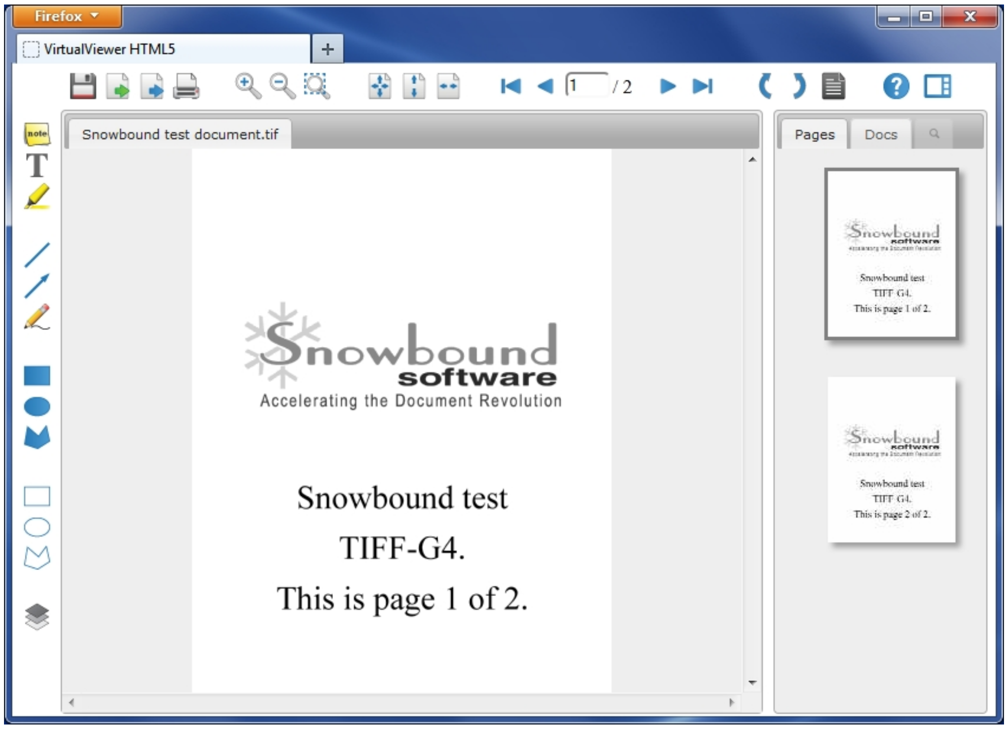
Running VirtualViewer HTML5 in a Browser

Once all components have been installed, VirtualViewer HTML5 will start up from any supported browser. No client components are needed on the client machine.

To start VirtualViewer HTML5, open your .html file in a browser. For example, open index.html.

Note:
Make sure that your web service such as Tomcat is running when you open the viewer in your browser.

The following example shows VirtualViewer HTML5 loaded in a browser:



Verifying that the Sample Documents Work in VirtualViewer HTML5

Snowbound Software provides sample documents in the VirtualViewer HTML5 installation to get you started. The sample files are located in the Sample-Documents subdirectory. The web.xml file delivered with VirtualViewer HTML5 located in `VirtualViewerHTML5\WEB-INF` specifies the Sample-Documents subdirectory as the default location of the sample files in the `filePath` parameter.

To view the sample documents, enter the URL shown in the **Displaying the Sample Documents** example below. Replace **server** and **port** with your server and port where shown in the example. To view any of the sample documents, specify the document name at the end of the URL after `documentId` with the document name.

Example 1.1: Displaying the Sample Documents

```
http://localhost:8080/VirtualViewerHTML5/index.html?documentId=Snowboun-
d Test Document.tif
```

If you are able to see all of the documents that came in the Sample-Documents directory of your VirtualViewer HTML5 installation, then you have successfully installed it. To view the other documents specify the filename after the `documentId` in the URL. If you are not able to see the documents in the viewer, please see the Appendix F, Troubleshooting section. If you are still not able to see the documents, please file a ticket with Snowbound Support at http://support.snowbound.com.

Verifying that Your Documents Work in VirtualViewer HTML5

Now you can move on to viewing your documents by placing them in the Sample-Documents directory and then specifying the document's file name after the `documentId` in the URL.

For example, if you want to display the file named **test.tif**, add that file to your Sample-Documents directory and **test.tif** after `documentId` as shown in the following example:

Example 1.2: Specifying the Document to Display

```
http://localhost:8080/VirtualViewerHTML5/index.html?documentId=filename-
.ext

http://localhost:8080/VirtualViewerHTML5/index.html?documentId=test.tif
```

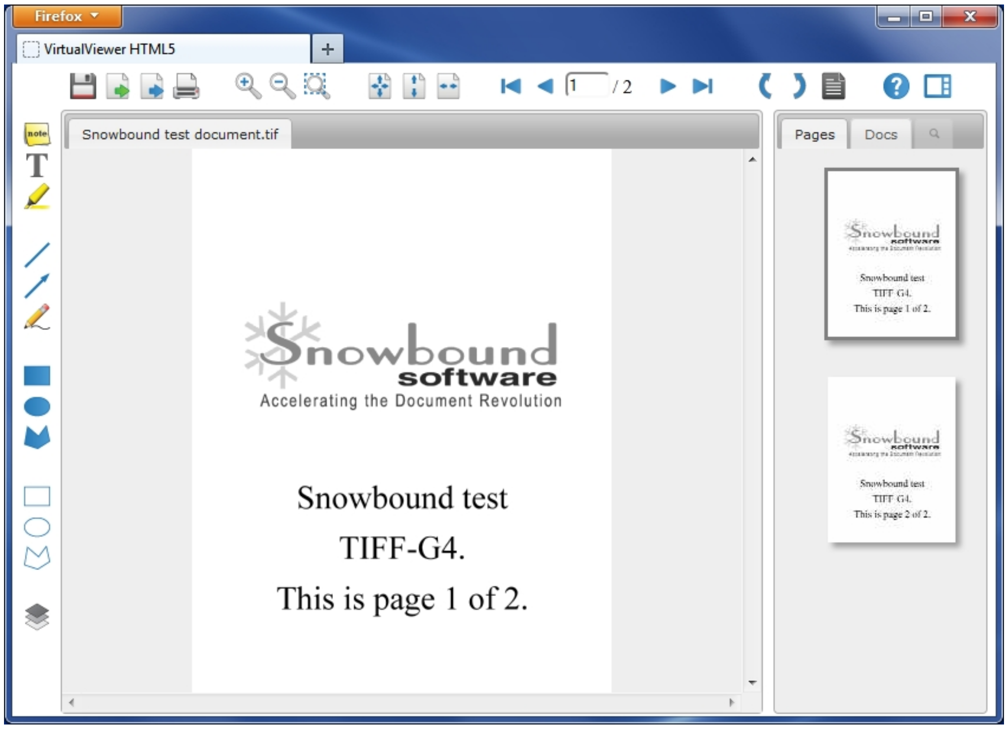
The `documentId` should be a filename if the default content handler is used. Otherwise, it can be whatever the custom content handler expects for a `documentId`. For more information, please see Connecting to Your Document Store.

For information on configuring VirtualViewer HTML5, please see Chapter 3, Customizing the Configuration .

Please see the next topic Chapter 2, Using VirtualViewer HTML5 Client.

Chapter 2 - Using VirtualViewer HTML5

This chapter describes the available functionality and features in VirtualViewer HTML5.



Snowbound software
Accelerating the Document Revolution

Snowbound test
TIFF-G4.
This is page 1 of 2.

The Annotation Toolbar

Creating Annotations

To create annotations, click on the annotation to select it and then click and drag your mouse on the document. Release the mouse when you are done drawing the annotation. The available annotation buttons are: sticky note, rubber band stamp text, highlighted rectangle, line, arrow, freehand, filled rectangle, filled ellipse, filled polygon, rectangle, ellipse, and polygon.

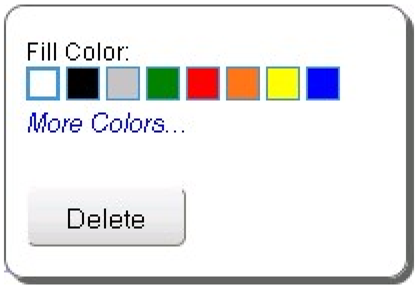
Note:
Annotations are not supported on the iPhone and iPad platforms.

To display a contextual annotation box, click on the annotation and then right-click on your mouse. The contextual annotation box allows you to:

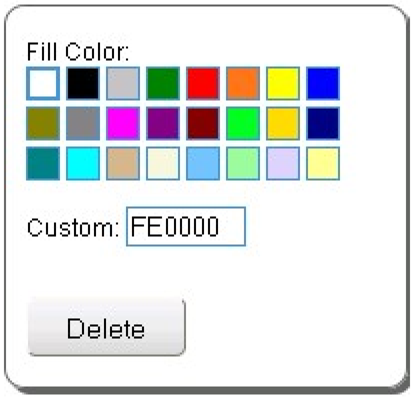
- Select a color to fill in the annotation.
- Select a line color.
- Adjust the line size for a line annotation.
- Edit the text for a text annotation.

Editing a Filled Annotation

To select the fill color for a filled annotation, right-click on the annotation. In the contextual annotation box, select the **fill color**.



To display more fill colors, select the **More Colors...** link. The Fill Color box expands to display more colors to select from:

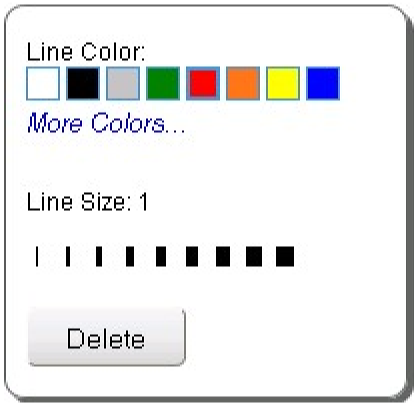


Custom: FE0000

In the Custom: field, you can enter a customized color code as the Red Green Blue (RGB) color code. For example, for the color red, enter the customized RGB color code of FE0000.

Editing a Line Annotation

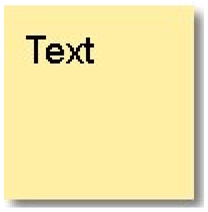
To adjust the line color in a line annotation, right-click on the annotation. In the contextual annotation box, select a **line color**. To display more fill colors, select the **More Colors...** link. The Fill Color box expands to display more colors to select from.



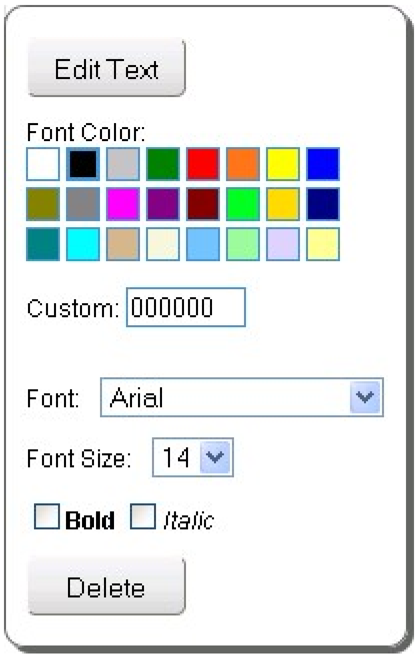
To adjust the line size, right-click on the line annotation. In the contextual annotation box, select the **line size** from the available line weights of 1 to 9.

Editing a Text Annotation

Select the Sticky Note text annotation.



Right-click on the text annotation to open the annotation contextual box.



To adjust the text color in a text annotation, select a **text color**. To display more fill colors, select the **More Colors...** link. The Fill Color box expands to display more colors to select from.

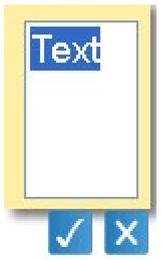
In the Custom: field, you can enter a **customized color code**.

In the Font field, select the **font** that you would like for the text.

In the Font Size drop down box, select the **font size** for the text.

Select the **Bold checkbox** for bold text. Select the **Italic checkbox** for Italic text.

Select the **Edit Text** button to edit the text in the text annotation.



Type in the new text in the highlighted text annotation box. Select the checkmark to accept the changed text. Select the X box to keep the text the same.

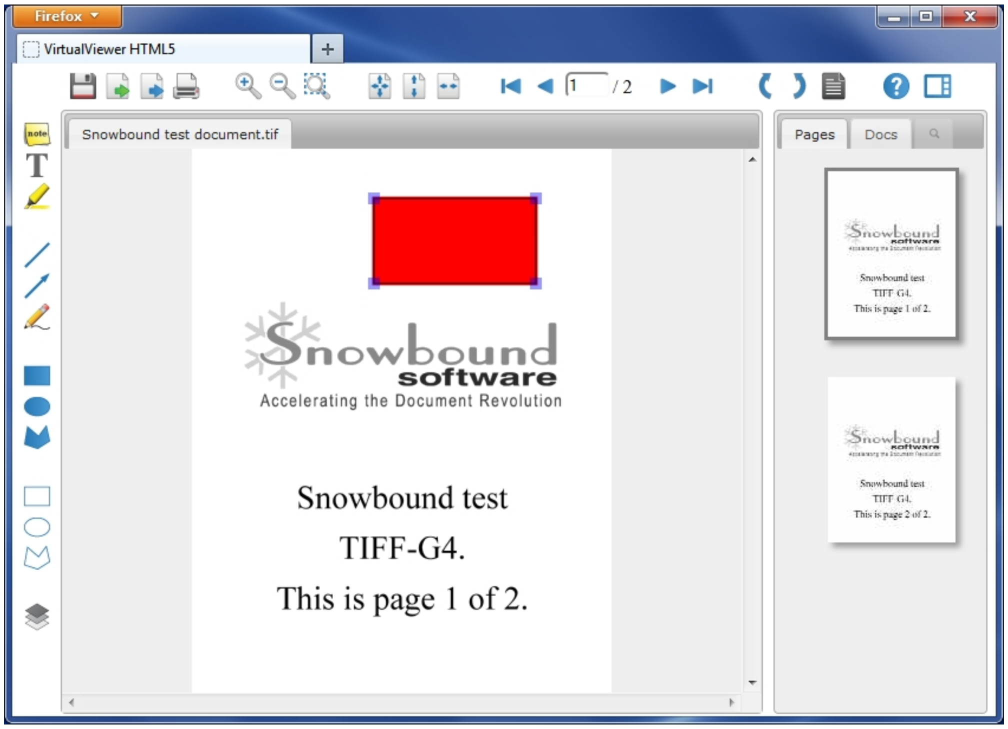
Moving an Annotation

To move an annotation, click on it until it is highlighted and selection squares display on each of the annotation’s corners. Drag the highlighted annotation until it is in the proper location.

Resizing an Annotation

To resize an annotation, click on it until it is highlighted and selection squares display on each of the annotation’s corners. Drag one of the selection squares, except for the top left one, to resize the annotation to the desired size. The following is the expected behavior for the highlighted annotation and selection squares:

- Select the **top left selection square** to drag the annotation to a new location. Dragging on other non-selection square areas of the annotation sets the upper left selection square under the mouse pointer.
- Select any of the **other selection squares** other than the top left one to resize the annotation.



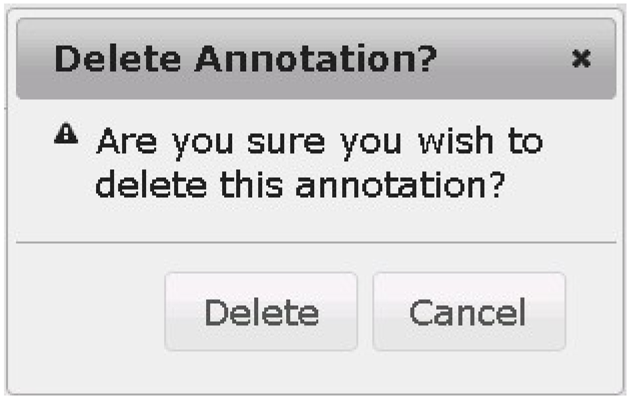
Saving Annotations

To save annotations, select the **Save Document** button.

Deleting Annotations

To delete an annotation, right-click on the annotation to display the contextual annotation box.

In the Delete Annotation box, select the **Delete** button to delete the annotation.


Delete Annotation?

Are you sure you wish to delete this annotation?

Undo a Deleted Annotation

As of VirtualViewer HTML5 V3.0, the undo a deleted annotation is no longer available.

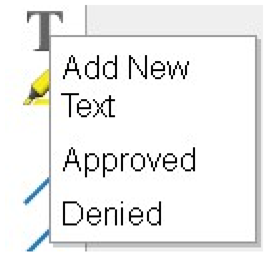
Using Rubber Stamp Annotation Functionality

A Rubber Stamp is a text annotation with pre-defined text that may also contain pre-defined font characteristics. Your system administrator has the ability to define a list of pre-configured Rubber Stamps through the `enableRubberStamp` parameter in the config.js file. For more information on configuring rubber stamp annotation functionality, please see Configuring Rubber Stamp Annotation Functionality in Chapter 3, Customizing the Configuration.

If the `enableRubberStamp` parameter is set to true and one or more Rubber Stamps are defined, then clicking on the **Text Edit** annotation toolbar button as shown below will produce the rubber stamp text menu.

T

This menu allows you to select a Rubber Stamp from the available options or to Add New Text to add a traditional text annotation.



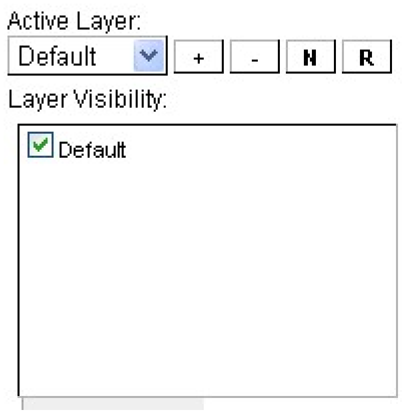
Note:
If the `enableRubberStamp` parameter is set to false, then clicking the Text Edit annotation button allows you to select only Add New Text to add a text annotation.

Using the Layer Manager

Annotations are placed on *layers*. The layer can be used to determine who can see the annotations on that layer (permissions) when printed or viewed. Layers can also be used to manage redactions. You may do the following:

- Put all the annotations on one layer so everyone has the same permission to view, write, and print the annotations.
- Create a layer for each annotation (FileNet style).
- Create a layer for each user making annotations.
- Create one or more redaction layers so that information in the document is obscured by an annotation that is *burned in* and cannot be removed.

To use the layer manager, select the Layer Manager button.



The Active Layer Window is displayed.

Note:
If no layers exist, a default layer is present.

From the drop down list, select the name of the layer that you want visible.

The active layers display with checkmarks.

Creating a New Layer

To create a new layer, select the plus button.

In the dialog box, enter the name of the new layer.

Note:
The layer name is limited to 50 standard characters.

The layer that you added displays as an active layer.

Deleting a Layer

To delete a layer, select the minus button.

VirtualViewer HTML5 displays a message asking “Are you sure that you want to delete the layer?”

Select **OK** to delete the layer.

Renaming a Layer

To rename a layer, select the **N** button.

In the dialog box, enter the new name for the layer.

The Active Layer window displays with the new layer name.

Redacting a Layer

A redaction layer is created just as any other annotation layer. Any objects on the layer will be burned in if upon retrieval the layer is given the Redaction permission. To create a redaction layer, select the **R** button.

Printing Layers

When printing a document, you may choose to print with or without annotations.

For more information on configuring the Print dialog box to display the Include Annotations checkbox, please see Print Dialog Box: Displaying the Include Annotations Checkbox in Chapter 3, Customizing the Configuration.

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

The Page and Document Toolbar

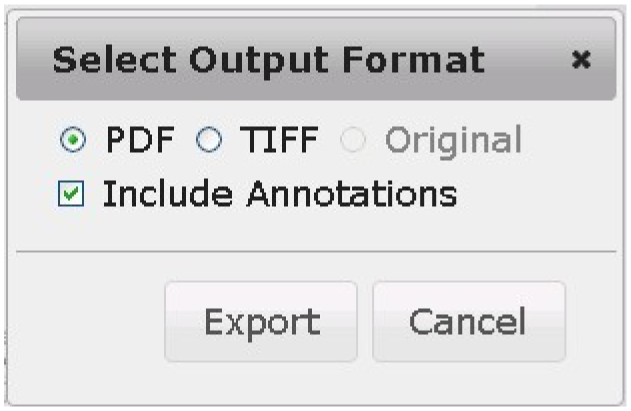
Exporting a Document

To export a document, select the **Export Document** button . The Export Document function allows regular and virtual documents to be exported.

Exporting a Document with Annotations

The Export dialog box contains the Include Annotations checkbox to select the option to export a document with annotations. Annotations will only be included when the **Include Annotations** checkbox is selected. The default is set to not include annotations when exporting. When exporting with annotations, only the visible layers are included. When the Include Annotations checkbox is selected, the option to export the file as Original will be disabled. The Include Annotations checkbox is only supported when either the PDF or TIFF format is checked. To export the file as Original, un-check Include Annotations to enable and make available the option for Original. Select the **Export** button to export.

For more information on configuring the Export dialog box to display the Include Annotations checkbox, please see Export Dialog Box: Displaying the Include Annotations Checkbox in Chapter 3, Customizing the Configuration.



Select Output Format

[✓ PDF] TIFF Original

[✓ Include Annotations]

Sending a Document

To send a document, select the **Send Document** button .

For more information on configuring the Send Document functions, please see Using the Send Document Functions in Chapter 3, Customizing the Configuration.

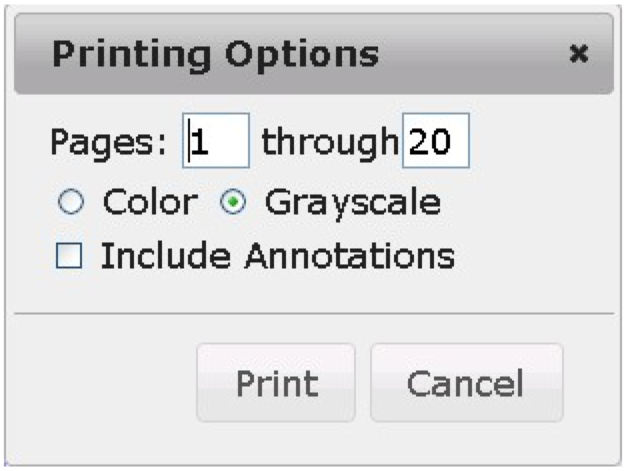
Printing

To print, select the **Print** button.

Printing with or without Annotations

The print dialog box contains the Include Annotations checkbox to select the option to print with or without annotations. Annotations will only be included when the **Included Annotations** checkbox is selected. The default is set to not include annotations when printing. When printing with annotations, only the visible layers are included.

For more information on configuring the Print dialog box to display the Include Annotations checkbox, please see Displaying the Include Annotations Checkbox in Chapter 3, Customizing the Configuration.


Printing Options ×

Pages: 1 through 20

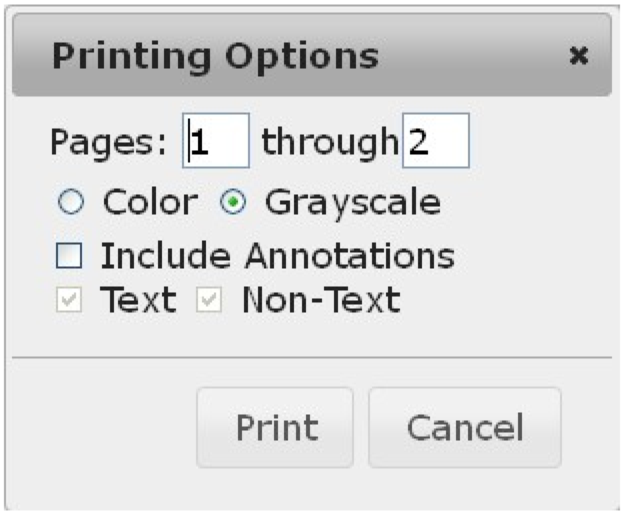
○ Color [✓ Grayscale]

☐ Include Annotations

Printing Text and Non-text Annotations Separately

If your version of VirtualViewer HTML5 is configured for it, the print dialog box may also contain the Text and Non-text checkbox to print text and non-text annotations separately. Text annotations will be printed separately when the **Text** checkbox is selected. Non-text annotations will be printed separately when the **Non-text** checkbox is selected. The default is to not include the ability to print Text and Non-text annotations separately.

For more information on configuring the Print dialog box to display the Text and Non-text checkbox, please see Displaying the Text and Non-text Checkbox in Chapter 3, Customizing the Configuration.


Printing Options ×

Pages: 1 through 2

○ Color [✓ Grayscale]

☐ Include Annotations

[✓ Text] [✓ Non-Text]

Zooming

To zoom, select one of the **Zooming Controls** buttons. The available Zooming Controls buttons are:

Zoom In and Zoom Out .

Rubber Band Zoom

To use rubber band zoom, select the **Rubber Band Zoom** button and then drag your mouse to select the area that you want to zoom in on.

Fit-to-Page

To fit the document to the page, select one of the **Fit-to Controls** buttons. The available Fit-to Controls buttons are:

Fit-to-page , Fit-to-width , and Fit-to-height .

Page Controls

To move from page to page, select one of **Page Controls** buttons. The available Page Controls buttons are:

First Page , Previous Page , Next Page , and Last Page .

Page Manipulation

To manipulate the pages, select one of **Page Manipulation** buttons. The available Page Manipulation buttons are:

Rotate Clockwise and Rotate Counter-clockwise .

Inverting

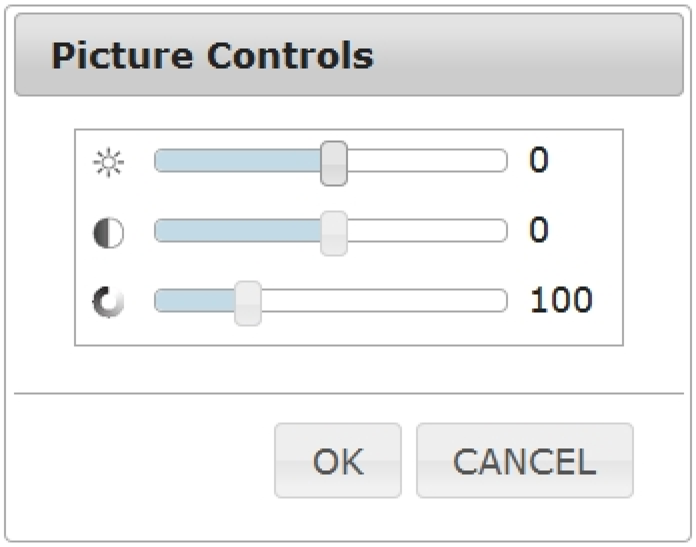
To invert the document, select the **Invert** button .

Note:
As of V3.3, the Invert button does not appear on the main toolbar. It has been replaced by the Pictures Controls button. To turn the Invert button back on, use the invertImage() API method as explained in Customizing VirtualViewer HTML5 through JavaScript API Methods.

Picture Controls

To adjust image properties (picture controls), select the **Picture Controls** button .

Once the Picture Controls button is selected, VirtualViewer HTML5 displays the Picture Controls window.



You can adjust the Brightness, Contrast, and Gamma by sliding the control bar to increase or decrease the brightness, contrast, and gamma.

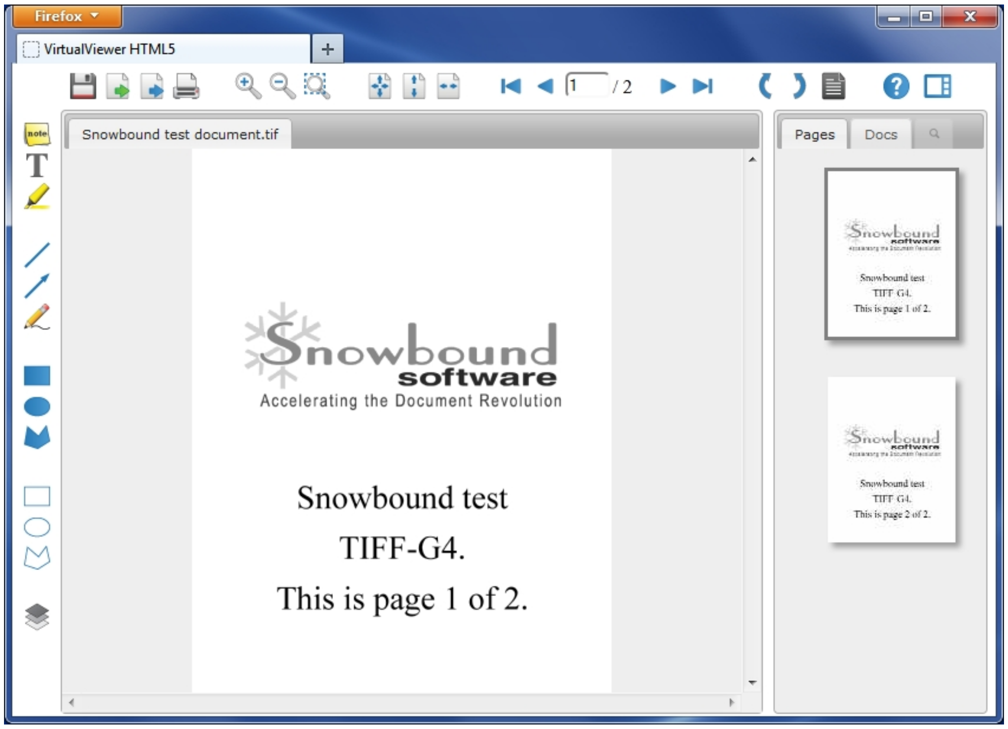
Picture Controls are measured on a range of -125 to 125.

Changes made to the Picture Controls properties are page specific and only applied to the page actively in focus.

Changes made to the Picture Controls properties will be seen in the viewer, in near real time, as the adjustments are made.

The Pages and Documents Panels

The panel on the right side of the screen shows the thumbnails for the current image and for all the documents made available by multiple documents mode. Select the **Pages** tab to display the thumbnails for the current image being viewed. Select the **Documents** tab to display thumbnails for the first page of every document made available by multiple documents mode.



To select a specific page or document simply click on the corresponding thumbnail and that page or document will load into the main viewing area.

Hiding the Pages and Documents Panel

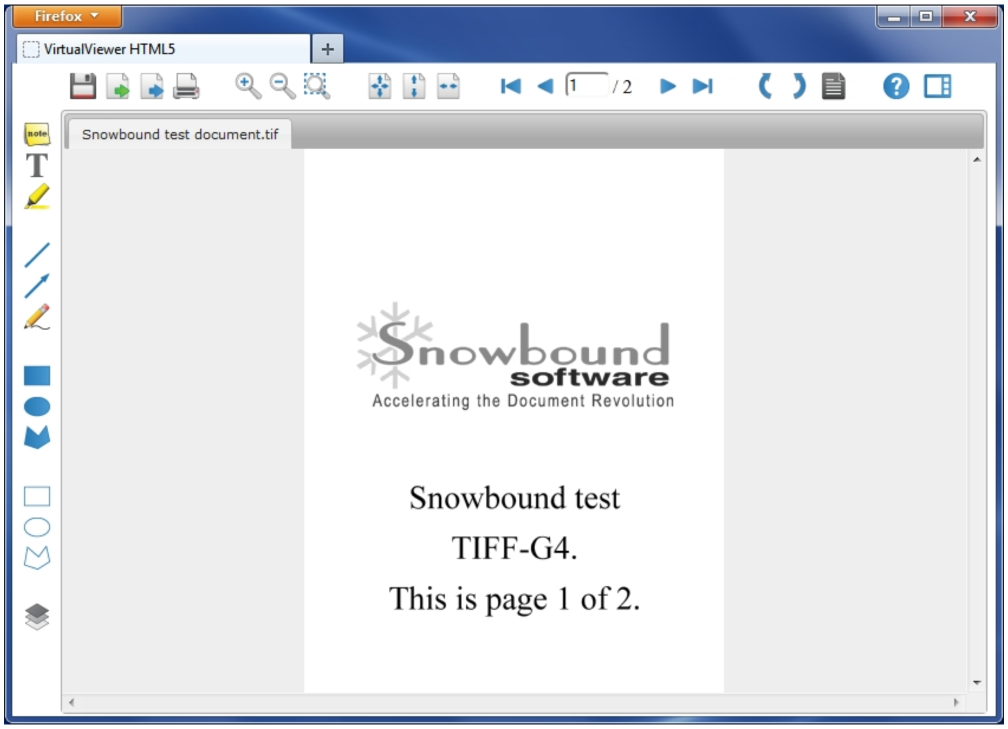
The Thumbnail panel provides a convenient way to:

- Navigate to any page in a document in the Pages panel.
- Select another document to view from the multiple Documents panel.
- Create a new document by dragging and dropping pages from another document.

However, this convenience does have a price. VirtualViewer HTML5 performance degrades because it is processing every page in the document Pages panel and/or the first page of every document in the Documents panel. If you want to speed up performance, you may want to disable or hide the thumbnail navigation panels. For more information on disabling or hiding the thumbnail panel, please see Hiding the Thumbnail Panel in Chapter 3, Customizing the Configuration.

To hide or show the Pages and Documents panel, select the **toggle** button .

The following shows VirtualViewer HTML5 with the Thumbnail Panel hidden:



Manipulating Page Order using Thumbnails

VirtualViewer HTML5 allows you to add, remove and reorder pages by cutting and pasting the page thumbnails. This section describes how to enable and use the Page Manipulations feature.

Page Manipulations

Page manipulations are enabled by default. For more information on disabling page manipulations, please see Disabling Page Manipulations in Chapter 3, Customizing the Configuration.

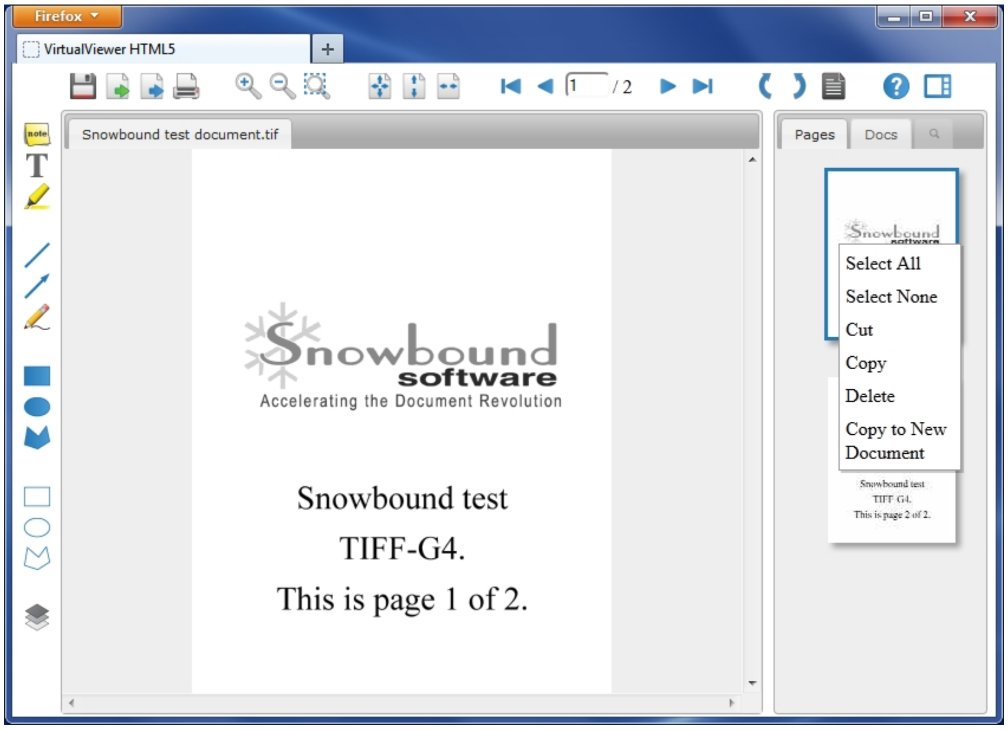
Selecting a Page

To select a page for page manipulation, left click on a page thumbnail in the Pages tab. A gray selection border around the thumbnail indicates that it has been selected for page manipulation.

- Hold the **Ctrl** key while selecting multiple page thumbnails to allow the selection of all thumbnails selected for page manipulation.
- Hold the **Shift** key and select a single thumbnail while one or more thumbnails are already selected to highlight all pages between the highest page selected before the new selection.

Loading the Page Manipulation Context Menu

Right-click on a page thumbnail to load the page manipulation context menu.



Cutting, Copying, Deleting and Inserting Pages

You can cut, copy, delete and insert a page from one document into another document open in the same instance of VirtualViewer HTML5.

Note:
Drag and drop functionality is not supported. You cannot insert pages between two separate instances of VirtualViewer HTML5.

Saving Page Manipulations

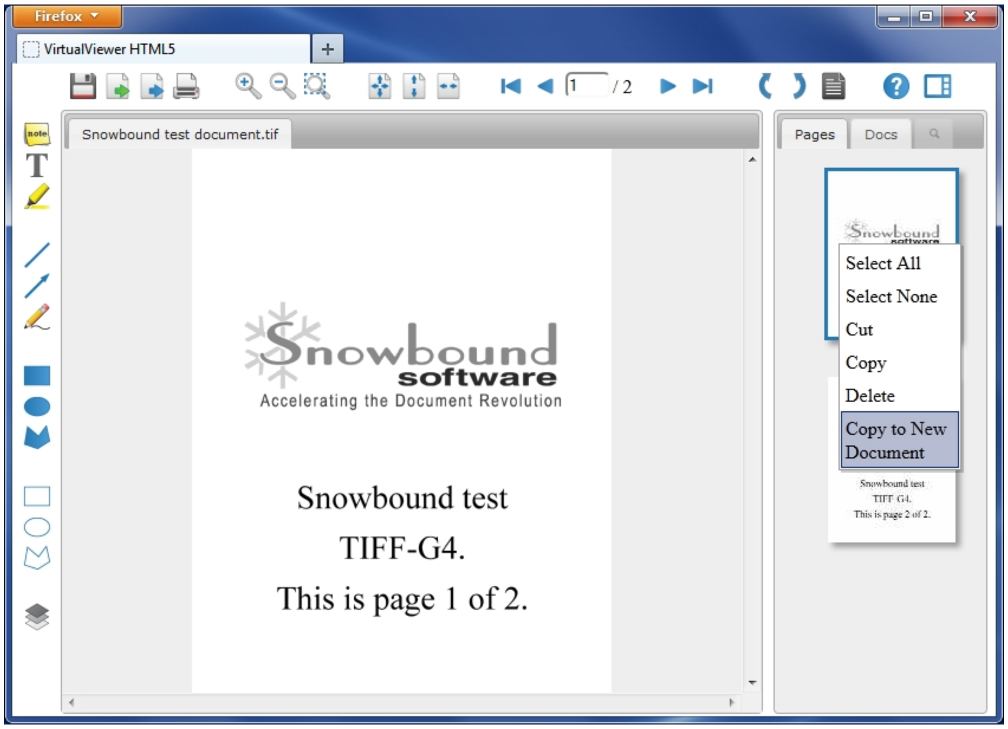
Select **Save** to save page manipulations, including rotations and inversions, to the file currently being viewed.

Copy to New Document

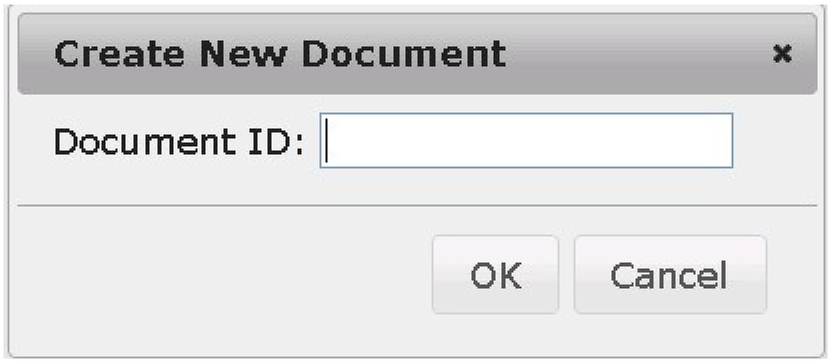
To copy to a new document, follow the steps below:

1. Click on the **page thumbnail** or **page thumbnails** that you want to copy to the new document.

2. Right-click on the page thumbnail(s) to load the page manipulation context menu. Select **Copy to New Document** from the Page Manipulations menu.



3. In the Create New Document window, enter the new document name in the Document ID field and select the **OK** button.



The new document is displayed in a tab with the document name that you entered. It contains the pages that you selected.

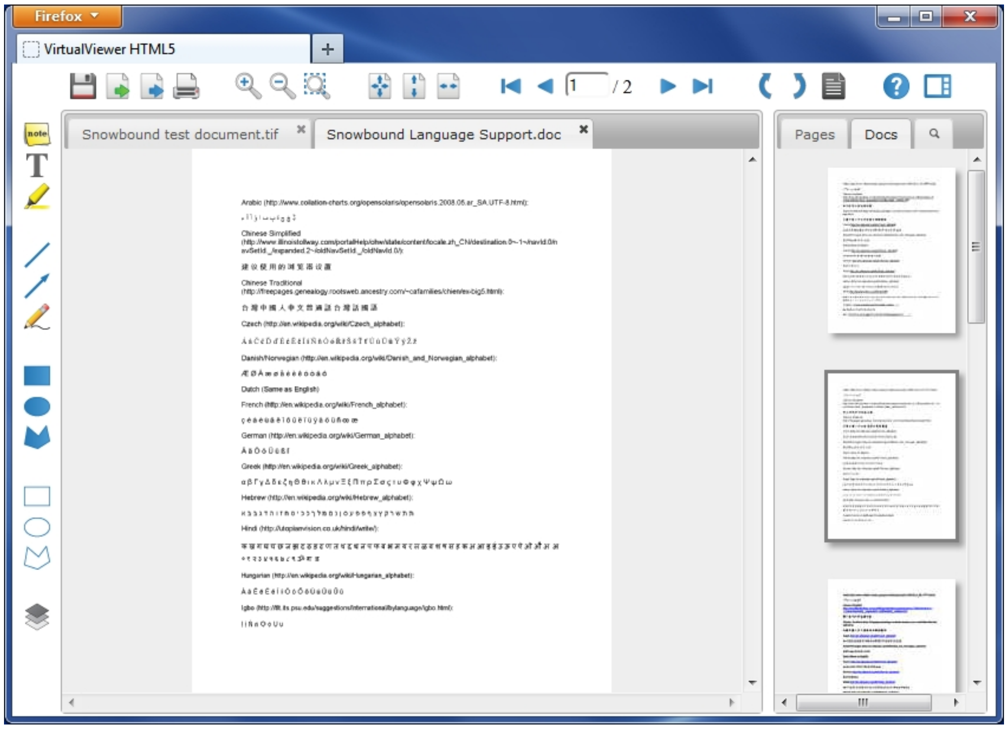
For more information on configuring Copy to New Document, please see Disabling Copy to New Document in Chapter 3, Customizing the Configuration .

Open Multiple Documents

To open multiple documents, select the **Documents** tab located at the top of the thumbnail panel. From the Documents pane, left-click any document you would like to open in the main viewer. When a new document has been clicked in Documents pane, it will display in the main viewer and have a new document tab created for it. Select document tabs to display any open documents.

Please see Configuring the Document Thumbnail Panel Display in Chapter 3, Customizing the Configuration for more information on configuring the `multipleDocMode` parameter.

If your documents load slowly in multiple documents mode, please see Documents Slowly to Load in Multiple Documents Mode in Appendix F, Troubleshooting.



Text Searching

The text searching tab is enabled by default. The text searching tab is enabled by setting the `showSearch` config.js parameter to true. Set the parameter to false to disable the text searching tab. Please see the example below:

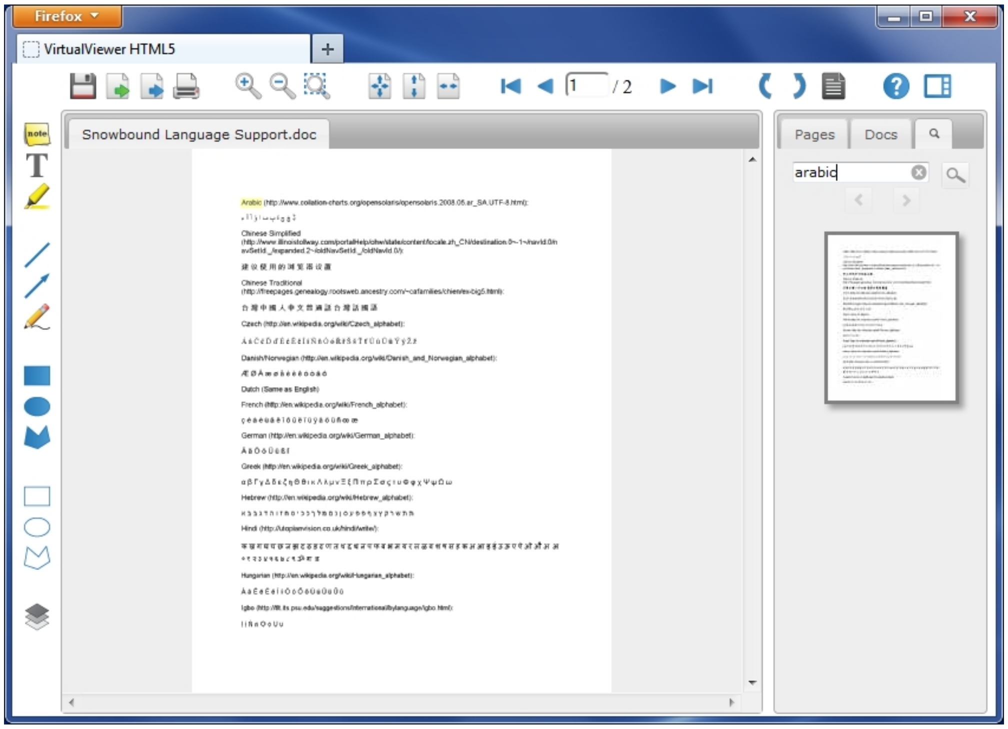
Example 2.1: Enabling Text Searching

```
var showSearch= true;
```

To determine whether or not text searches should be case sensitive, set the `searchCaseSensitive` config.js parameter to true or false depending if you want case sensitivity turned on or off. The default value is false. See the example below:

Example 2.2: Enabling Case Sensitivity in Text Searches

```
var searchCaseSensitive = false;
```



Arabic (http://www.collation-charts.org/opensolaris/opensolaris.2008.05.ar_SA.UTF-8.html):

[illegible]

Chinese Simplified
(http://www.illinoistollway.com/portalHelp/ohw/state/content/locale.zh_CN/destination.0~-1~/navId.0/navSetId._/expanded.2~/oldNavSetId._/oldNavId.0/):

建 议 使 用 的 浏 览 器 设 置

Chinese Traditional
(http://freepages.genealogy.rootsweb.ancestry.com/~cafamilies/chien/ex-big5.html):

台 灣 中 國 人 中 文 普 通 話 台 灣 話 國 語

Czech (http://en.wikipedia.org/wiki/Czech_alphabet):

Á á Č č Ď ď É é Ě ě Í í Ň ň Ó ó Ř ř Š š Ť ť Ú ú Ů ů Ý ý Ž ž

Danish/Norwegian (http://en.wikipedia.org/wiki/Danish_and_Norwegian_alphabet):

Æ Ø Å æ ø å é è ê ó ò ô â ó

Dutch (Same as English)

French (http://en.wikipedia.org/wiki/French_alphabet):

ç é â ê û à è î ô ù ë ï ü ÿ ä ö ü ñ œ æ

German (http://en.wikipedia.org/wiki/German_alphabet):

Ä ä Ö ö Ü ü ß ſ

Greek (http://en.wikipedia.org/wiki/Greek_alphabet):

α β Γ γ Δ δ ε ζ η Θ θ ι κ Λ λ μ ν Ξ ξ Π π ρ Σ σ ς τ υ Φ φ χ Ψ ψ Ω ω

Hebrew (http://en.wikipedia.org/wiki/Hebrew_alphabet):

א ב ב ג ג ד ה ו ז ח ט י כ ך ל מ ם נ ן ס ע פ פ ף צ ץ ק ר ש ת ת

Hindi (http://utopianvision.co.uk/hindi/write/):

[illegible]

Hungarian (http://en.wikipedia.org/wiki/Hungarian_alphabet):

Á á É é É é Í í Ó ó Ő ő Ú ú Ű ű Ü ü

Igbo (http://tlt.its.psu.edu/suggestions/international/bylanguage/igbo.html):

Ị ị Ṅ ṅ Ọ ọ Ụ ụ

Setting the Default Colors

You can configure the default colors for the first and second search match by setting the values for the `searchDefaultColor` and the `searchSelectedColor` in the **vvDefines.js** file found in the js directory. Please see the following example:

Example 2.3: Setting the Default Colors

```
vvDefines = {
searchDefaultColor: "rgba(255,78,0,0.2)",
searchSelectedColor: "rgba(255,255,0,0.2)",
```

Please see the next topic Chapter 3, Customizing the Configuration.

Chapter 3 - Customizing the Configuration

This chapter shows how to configure VirtualViewer HTML5 on your system.

Configuring web.xml

The **web.xml** file contains a number of tags that define both servlets and their behavior. The web.xml file is located in the **VirtualViewerHTML5\WEB-INF** directory. There are two groups of tags. The first group is a pair of `<servlet>` tags, and the second group is a pair of `<servlet-mapping>` tags. All of these tags are now added by default to VirtualViewer HTML5 web.xml when the `contentServerType` parameter is set to integrated.

Retrieval Servlet

The first `<servlet>` is the Response Server, which is responsible for handling when data needs to be sent to VirtualViewer HTML5. Various parameters within its tag define where to retrieve documents from, how it should render and deliver them to VirtualViewer HTML5, how to cache documents, logging, and more.

Example 3.1: Retrieval Servlet

```
<servlet>
 <servlet-name>RetrievalServet</servlet-name>
 <servlet-class>
 com.snowbound.snapserv.servlet.ResponseServer
</servlet-class>
<init-param>
 <param-name>contentHandlerClass</param-name>
 <param-value>com.snowbound.snapserv.servlet.FileContentHandler
</param-value>
</init-param>
<init-param>
<param-name>logLevel</param-name>
<param-value>FINE</param-value>
</init-param>
</servlet>
```

Upload Servlet

The second `<servlet>` is the Request Server. It is responsible for handling when data needs to be sent from VirtualViewer HTML5. Various parameters within its tag define settings for the serlvet when saving documents and annotations.

Example 3.2: Upload Servlet

```
<servlet>
```

```
  <servlet-name>UploadServer</servlet-name>
<servlet-class>com.snowbound.snapserv.servlet.RequestServer</servlet-
class>
  <init-param>
   <param-name>saveAnnotationsAsXml</param-name>
   <param-value>false</param-value>
</init-param>
<init-param>
   <param-name>tmpDir</param-name>
   <param-value>c:/tmp/</param-value>
</init-param>
</servlet>
```

Defining the Servlet Paths

Each servlet has its own `<servlet-mapping>` tag to define the path it may be found. The default values should not be changed.

Example 3.3: Servlet Paths

```
<servlet-mapping>
  <servlet-name>RetrievalServlet</servlet-name>
<url-pattern>/ResponseServer</url-pattern>
 </servlet-mapping>
<servlet-mapping>
<servlet-name>UploadServlet</servlet-name>
<url-pattern>/RequestServer</url-pattern>
</servlet-mapping>
```

Display Your Documents

To view your own images in VirtualViewer HTML5, place them in the Sample-Documents directory and then specify the document's file name after the `documentId` in the URL.

For example, if you want to display the file named test.tif, add that file to your Sample-Documents directory and **test.tif** after `documentId` as shown in the following example:

Example 3.4: Specifying the Document to Display

```
http://localhost:8080/VirtualViewerHTML5/index.html?documentId=filename-
.ext

http://localhost:8080/VirtualViewerHTML5/index.html?documentId=test.tif
```

If you are not able to get your images to load, please submit a support ticket at http://support.snowbound.com or see the "Please wait while your image is loaded" Message Displays Indefinitely topic in Appendix F, Troubleshooting.

The `documentId` should be a filename if the default content handler is used, otherwise it can be whatever the custom content handler expects for a `documentId`. For more information, please see Connecting to Your Document Store.

Configuring config.js

You can configure the appearance of VirtualViewer HTML5 through the config.js file. This file is included with your installation in the VirtualViewerHTML5 directory. It allows you to configure colors, zoom levels, multiple documents mode, and error messages.

For example, to set the percentage to stop allowing users to zoom the image, set the `maxZoomPercent` parameter in the config.js file as shown in the following example:

```
var maxZoomPercent = 1000;
```

For a list of the available parameters that you can configure, please see Appendix A, Config.js Parameters.

Deploying on Multiple Machines or Directories

VirtualViewer HTML5 consists of two servers - an AJAX web application server and a Java Content Server. The evaluation version is configured in *Integrated Mode* which means both servers are in the same directory on the same machine. This is the optimal configuration for performance.

If your web server and content server need to be in different directories or on different machines, then you can configure VirtualViewer HTML5 in *HTTP Mode* using the `contentServerType` parameter in the web.xml file. The web.xml file is located in the VirtualViewerHTML5\WEB-INF directory. This mode adds an extra level of encryption between the servers so that the traffic can be sent over the network between the two servers.

Example 3.5: Setting contentServerType to HTTP

```
<init-param>
<param-name>contentServerType</param-name>
<param-value>http</param-value>
</init-param>
```

As of VirtualViewer HTML5 V3.0, you will need a special build to separate the servers. Please contact Snowbound Technical Support at http://support.snowbound.com and request a multi-server VirtualViewer HTML5 build.

Customizing the User Interface

VirtualViewer HTML5 can be customized in many ways. One of the most popular customizations is making it read-only.

We provide the VirtualViewer HTML5 client with almost all options turned on. It is easy to turn off options such as Save Document. Edit theindex.html file and comment out or remove the `saveDocument` item as shown in the example below:

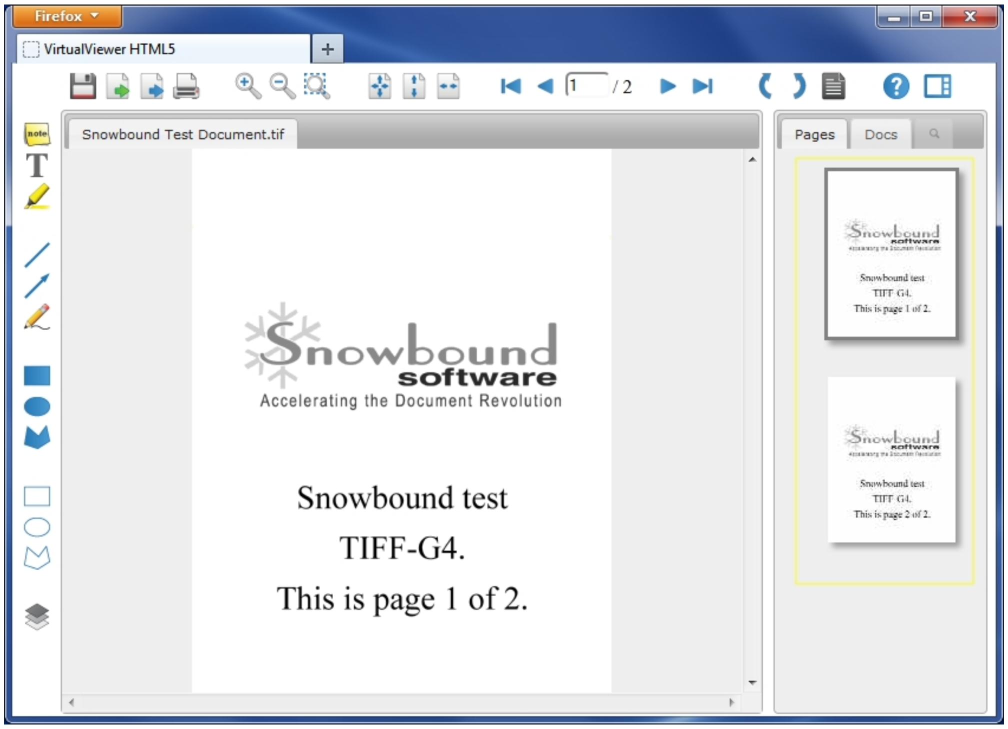
Example 3.6: Customizing What is Displayed in the VirtualViewer HTML5 Client

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

You can do this with other buttons and menus as well. The descriptions of the options are in Chapter 2, Using VirtualViewer HTML5.

Another trick is to have a different index.html for each type of user, or to have a script generate the HTML on the fly.

Configuring the Pages and Document Panel Display



Snowbound test

TIFF-G4.

This is page 1 of 2.

You can set the `multipleDocMode` parameter in the config.js file to configure which documents will be shown within the Documents pane of VirtualViewer HTML5. It can be also be used to limit what documents are available to the user.

Please see Appendix A, Config.js Parameters for more information on setting the `multipleDocMode` configuration parameter.

The `multipleDocMode` configuration parameter supports the following three values as options:

- availableDocuments
- viewedDocuments
- specifiedDocuments

Note:
Generating the thumbnails for a large number of documents can be a time consuming operation that will slow down performance. Please choose the document mode accordingly. If the number of documents is large (more than 100), then you may want to consider limiting the list by using `specifiedDocuments` mode.

availableDocuments

The `availableDocuments` option displays the documents that are available to the current user.

The connector to your document storage, the content handler, determines what documents are listed by returning them from its `getAvailableDocumentIds` call. Please see the `getAvailableDocumentIds()` method description in the Content Handler Methods in Chapter 4, Using Advanced Features.

The default content handler is the File Content Handler. It should return all of the documents in the document directory once `getAvailableDocumentIds` is implemented in the sample File Content Handler.

Example 3.7: Setting multipleDocMode to availableDocuments

This example shows how to set the **`multipleDocMode`** parameter in the config.js file to use **`availableDocuments`**.

```
var multipleDocMode = multipleDocModes.availableDocuments;
```

Documents handling when configured to use **`availableDocuments`**:

The **`getAvailableDocumentIds()`** method is called in the content handler to populate the list of documents. Please see the **`getAvailableDocumentIds()`** method description in the Content Handler Methods in Chapter 4, Using Advanced Features .

viewedDocuments

The `viewedDocuments` option adds documents to the set of documents as the user views them during the current session.

Example 3.8: Setting multipleDocMode to viewedDocuments

This example shows how to set the **`multipleDocMode`** parameter in the config.js file to use **`viewedDocuments`**.

```
var multipleDocMode = multipleDocModes.viewedDocuments;
```

Documents handling when configured to use **`viewedDocuments`**:

Documents are passed to the viewer via the URL `documentId` parameter:

```
index.html?documentId=filename
```

Documents are loaded into the viewer with the onload event:

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

specifiedDocuments

The `specifiedDocuments` option limits the documents available for viewing to those specified in an array.

Example 3.9: Changing multipleDocMode from availableDocuments to specifiedDocuments

This example shows how to change `multipleDocMode` from `availableDocuments` to `specifiedDocuments` with the set of specified documents limited to: help.doc, info.tif, image.jpg.

In the config.js file, change the value **`multipleDocMode`** to **`specifiedDocuments`** and add a new line defining the array of specified documents:

```
var multipleDocMode = multipleDocModes.specifiedDocuments;
var SD = new Array("help.doc","info.tif","image.jpg);
```

In the index.html file, change the value of the onload event.

Results in index.html:

```
<body onload="myFlexSnap.initSpecifiedDocuments(SD);">
```

Hiding the Pages and Documents Panel

The Pages and Documents panel provides a convenient way to:

- Navigate to any page in a document in the Pages panel.
- Select another document to view from the multiple Documents panel.
- Create a new document by dragging and dropping pages from another document.

However, this convenience does have a price. VirtualViewer HTML5 performance degrades because it is processing every page in the document Pages panel and/or the first page of every document in the Documents panel. If you want to speed up performance, you may want to disable or hide the Pages and Documents panel by setting the `showThumbnailPanel` parameter to false in the config.js file as shown in the example below:

```
var showThumbnailPanel = false;
```

Disabling Page Manipulations

Page manipulations are enabled by default. To disable page manipulations, the `pageManipulations` parameter must be set to false in the config.js file. This disables the Page Manipulations menu in VirtualViewer HTML5 and enables the Save Annotations menu choice in the File menu. To disable it, set the `pageManipulations` parameter to false in the config.js file as shown in the example below:

```
var pageManipulations = false;
```

Disabling Copy to New Document

The Copy to New Document functionality is enabled by default. To disable it, set the `pageManipulationsNewDocumentMenu` parameter to false in the config.js file as shown in the example below:

```
var pageManipulationsNewDocumentMenu = false;
```

Configuring Rubber Stamp Annotation Functionality

The Rubber Stamp functionality is enabled when the `enableRubberStamp` parameter is set to true and the config.js file contains one or more defined Rubber Stamps. The system will allow for a limited number of Rubber Stamps with the upper limit of available Rubber Stamps set at ten. To disable this functionality, set the `enableRubberStamp` parameter to false in the config.js file as in the example below:

```
var enableRubberStamp = false;
```

The system administrator has the ability to set the following pre-defined font characteristics for Rubber Stamps:

- Font Face (Helvetica, Times New Roman, Arial, Courier, Courier New)
- Font Size (Any valid integer in range of 2-176)
- Font Color (Any valid HTML color code, specified in hexadecimal)
- Font Attributes (Normal/Bold/Italic)

Please see the following example for how we configure the two Rubber Stamps **Approved** and **Denied**:

Example 3.10: Configuring the Approved and Denied Rubber Stamps

```
var rubberStamp = [
{ textString: "Approved",
fontFace: "Times New Roman",
fontSize: 30,
fontBold: true,
fontItalic: true,
fontColor: "00FF00" }
{ textString: "Denied",
fontColor: "FF0000" }
];
```

Any font characteristics not defined by the system administrator will use the following default system characteristics:

- Font Face: Arial
- Font Size: 12
- Font Color: #FF0000
- Font Attributes: Normal

Configuring Default Annotation Values

The default appearance for a text annotation looks like a yellow sticky note. If you prefer a different look, the `annotationDefaults` config.js configuration parameter sets the default and is customizable.

Please see the following example:

Example 3.11: Configuring the Default Annotation Values

```
// Default appearance options for annotations
annotationDefaults: {
lineColor: "FE0000",
 lineWidth: 3,

 fillColor: "FE0000",
 stickyFillColor: "FCEFA1", // yellowish
stickyMargin: 10, // also need to adjust .vvStickyNote in webviewer.css


highlightFillColor: "FCEFA1",
highlightOpacity: 0.4,

 textString: "Text",

fontFace: "Arial",
 fontSize: 14,
 fontBold: false,
fontItalic: false,
fontStrike: false, // for future use
 fontUnderline: false, // for future use
fontColor: "000000"
 }
```

The system administrator has the ability to set the following default values for annotations:

- Line color
- Line width
- Fill color
- Sticky note fill color
- Sticky note margin

- Text string
- Font face
- Font size
- Font bold
- Font Italic
- Font strike

Displaying the Include Annotations Checkbox

Displaying the Include Annotations Checkbox

To display the Include Annotations checkbox in the Export dialog box, set the `exportBurnAnnotations` parameter to true in the config.js file as in the example below:

```
var exportBurnAnnotations = true;
```

Print Dialog Box

Displaying the Include Annotations Checkbox

To display the Include Annotations checkbox in the Print dialog box, set the `printBurnAnnotations` parameter to true in the config.js file as in the example below:

```
var printBurnAnnotations= true;
```

Displaying the Text and Non-text Checkbox

To display the Text and Non-text checkbox in the Print dialog box to print text and non-text annotations separately, set the `printShowTypeToggles` parameter to true in the config.js file as in the example below:

```
var printShowTypeToggles = true;
```

Using the Send Document Functions

If you wish to use the Send Document functions, you must implement the `sendDocumentContent` method in the content handler. You can implement it to do what you need with the document, such as send it to another process in your company's workflow.

The `hasSendDocument` parameter must be set to true to use Send Document as shown in the following example:

```
<param name=hasSendDocument value=true>
```

If you choose **Send Document** from the menu, it will pass just the document to the method.

If you choose **Send Document with Annotations** from the menu, it will pass the document with all visible annotation layers burned in to the method.

A separate Email command uses the following configuration parameters in the web.xml file located in the WEB-INF directory:

```
<param name=hasEmailDocument value=true>
```

```
<param name=emailSMTPServer value=mailserver.yourcompany.com>
```

```
<param name=emailFromAddress value=CustomerService@yourcompany.com>
```

Turning on Redaction Support

To turn on redaction support, set the following `supportRedactions` parameter to true in the content server web.config file. The default value is false.

Example 3.12: Turning on Redaction Support

```
<InitParams>
 <add key="supportRedactions" value="true"/>
  </InitParams>
```

Localization

VirtualViewer HTML5 localization supports auto detecting the language settings the user’s browser is configured to use. It then looks for a localization file in that language. If a localization file for the corresponding language exists, it will be used to display terms throughout the UI in that language.

For more information on setting language preferences in a browser, please see the following:

http://www.w3.org/International/questions/qa-lang-priorities.en.php

Localization Files

Localized files are stored in the following directory:

```
../VirtualViewerHTML5/resources/locale/
```

The default file, named **vv-zz.json**, located in that directory is a template for the mapping of the available English terms that can be localized.

The naming of the localized files should follow the syntax of **vv-zz.json**, replacing **zz** with the letter code of the language used for the appropriate translation. The letter codes follow the ISO 639 code values.

The Example Language Codes table shows languages and their language code to use.

Table 3.1: Example Language Codes

Language Name	Code
German	de
Spanish	es
French	fr
Hindi	hi
Japanese	ja
Portuguese	pt
Chinese	zh

Please visit the following links for additional resources on language codes:

http://www.loc.gov/standards/iso639-2/php/code_list.php

http://en.wikipedia.org/wiki/List_of_ISO_639-2_codes

Converting Terms

The terms that are displayed in **vv-zz.json** using all caps represent where the language specific replacements should be placed.

Each term includes a replacement text for the **alt** value and the **title** value, although these are most likely going to match each other.

The **alt** and the **title** values represent the displayed text that is shown if the image fails to load or when the user hovers the mouse over the image. It can describe the icon, or in the case of VirtualViewer HTML5, what action is associated with the corresponding icon.

Supporting Accents/Special Characters

Note:
To support the translation of terms to languages that use accents or special characters, these accents/special characters must first be converted to Unicode before including it in the translation file. You may also translate the entire string to Unicode, rather than just the accents/special characters.

Please visit the following links for additional resources to convert text to Unicode:

http://www.pinyin.info/tools/converter/chars2uninumbers.html

http://tokira.net/unicode/index.php

Example 3.13: Creating a French Language Translation File

The two letter code for **French** is **fr**.

Create a copy **vv-zz.json** and replace the **zz** with **fr**, resulting in a file named: **vv-fr.json**

To modify the display text for the **Rotate Left** button, look for the corresponding value:

```
"rotateLeft": {
  "alt":"ROTATELEFT.ALT",
 "title": "ROTATELEFT.TITLE"
},
```

* In this case, we will use the same value for both the alt and title values.

The **French** translation for Rotate Left is Rotation À Gauche.

Converting the accents/special characters in this translation into Unicode results in: Rotation &#192; Gauch

Using the converted results in **vv-fr.json** with:

```
"rotateLeft": {
 "alt": "Rotation  &#192; Gauche",
 "title": "Rotation  &#192; Gauche"
},
```

Force a Specific Language

If you do not wish to use the language settings auto-detection, you can force override the UI to use a specified translation.

This setting is controlled via a setting `localizeOptions` in vvDefines.js as shown in the example below. The vvDefines.js file is located in the following directory:

```
..VirtualViewerHTML5/js/ vvDefines.js
```

Example 3.14: Force a Specific Language

Remove the backslashes // before the word **language** and replace the values **zz** with the letter codes of the language file you want to force. Have the translation file available for reference.

```
localizeOptions: {
//language: "zz",
pathPrefix: "resources/locale"
},
```

Customizing VirtualViewer HTML5 through JavaScript API Methods

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

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

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

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

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

Please see the example below:

Example 3.15: 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.

Improving Performance or Quality

One of the differences between raster and vector formats is that raster formats have specific DPI (dots per inch) and bit depths. Vector formats aren't inherently black and white or color, and while they typically have sizing in inches, there is nothing that says what DPI or bit depth to use when rendered as a raster image.

When the content server pulls out a page from a vector format document, it must render that page to a certain DPI and bit depth, as well as save that image as some format to be passed to the client for display. The particular settings are determined on a per format basis by three servlet parameters.

To improve the performance, you can save your files as black and white or grayscale. For example, if you are converting a PDF document, you can save the document in the TIFF_G4_FAX file format. This will make the file size smaller and improve performance. Please note that there is always a trade off between performance and quality. To improve performance, the quality of the image may be less. This is true whenever working with any imaging software.

Setting the Bit Depth - xxxBitDepth

This parameter determines what bit depth to use when converting the vector page. Valid settings for this format are 1 (for black & white, smaller) or 24 (for color, bigger). If any pages of the vector document might be in color, then the setting of 24 should be used, since there is no way to tell if a page might or might not contain color vector objects.

The example below shows how to set the bit depth parameters in the web.xml file. For a list of of web.xml parameters, please see Appendix B, Servlet Tags for web.xml.

Example 3.16: Setting the Bit Depth

```
<init-param>
 <param-name>docxBitDepth</param-name>
 <param-value>24</param-value>
</init-param>
```

The available bit depth parameters are shown in the table below:

Table 3.2: Bit Depth Parameter Values and Description

Parameter Name	Description
`bitDepth`	The default bits per pixel for decompression of formats not specified with individual parameters.
`docxBitDepth`	The bit depth to use for Word 2007 documents. Valid values are 1 or 24.
`iocaBitDepth`	The bit depth to use when decompressing IOCA pages. Valid values are 1 or 24.
`modcaBitDepth`	The bit depth to use when decompressing MO:DCA pages. Valid values are 1 or 24.
`pclBitDepth`	The bit depth to use when decompressing PCL pages. Valid values are 1 or 24.
`pdfBitDepth`	The bit depth to use when decompressing PDF pages. Valid values are 1 or 24.
`pptBitDepth`	The bit depth to use when decompressing PPT pages. Valid values are 1 or 24.
`wordBitDepth`	The bit depth to use when decompressing Word pages. Valid values are 1 or 24.
`xlsBitDepth`	The bit depth to use when decompressing XLS pages. Valid values are 1 or 24.

Setting the DPI - xxxDPI

This parameter determines how many DPI (dots per inch) should be used when converting a vector page. Typical settings for this parameter are 150, 200, or 300. The higher the DPI, the higher the quality of the image, but also the bigger the size, which means more processing on the server and larger page sizes across the network. The optimal setting for this varies by format, but 200 is usually good for black & white documents or text, and 300 for color images and more detailed documents. Even higher numbers can be used (400, 600) but it can seriously affect speed of processing and available resources.

The example below shows how to set the DPI parameters in the web.xml file. For a list of of web.xml parameters, please see Appendix B, Servlet Tags for web.xml.

Example 3.17: Setting the DPI

```
<init-param>
 <param-name>docxDPI</param-name>
 <param-value>200</param-value>
</init-param>
```

The available DPI parameters are shown in the table below:

Table 3.3: DPI Parameter Values and Description

Parameter Name	Description
`docxDPI`	The Dots Per Inch to use for Word 2007 documents.
`iocaDPI`	The Dots Per Inch to use when decompressing IOCA pages.
`modcaDPI`	The Dots Per Inch to use when decompressing MO:DCA pages.
`pclDPI`	The Dots Per Inch to use when decompressing PCL pages.
`pdfDPI`	The Dots Per Inch to use when decompressing PDF pages.
`pptDPI`	The Dots Per Inch to use when decompressing PPT pages.
`wordDPI`	The Dots Per Inch to use when decompressing Word pages.
`xlsDPI`	The Dots Per Inch to use when decompressing XLS pages.

Setting the Format - xxxFormat

This parameter determines which format the vector page will be rendered to for sending to the client. Valid values for this parameter are TIFF_G4_FAX (black & white, best for text documents, small size), JPEG (color, good for images, lesser quality for text, small size), TIFF_LZW (color or greyscale, good for documents with text and color elements), or PNG (color, better for text than JPEG, not as small).

By adjusting these parameters in various combinations, you can find the best settings for your environment, documents, and user load.

The example below shows how to set the format parameters in the web.xml file. For a list of web.xml parameters, please see Appendix B, Servlet Tags for web.xml.

Example 3.18: Setting the Format

```
<init-param>
 <param-name>docxFormat</param-name>
 <param-value>TIFF_LZW</param-value>
</init-param>
```

The available format parameters are shown in the table below:

Table 3.4: Format Parameter Values and Description

Parameter Name	Description
`docxFormat`	The format to convert Word 2007 documents to. Valid values should are TIFF_G4, JPEG, TIFF_LZW, PNG.
`iocaFormat`	The format to convert IOCA pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
`modcaFormat`	The format to convert MO:DCA pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
`pclFormat`	The format to convert PCL pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
`pdfFormat`	The format to convert PDF pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
`pptFormat`	The format to convert PPT pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG
`wordFormat`	The format to convert Word pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG. The bit depth to use when decompressing XLS pages. Valid values are 1 or 24.
`xlsFormat`	The format to convert XLS pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
`xlsDPI`	The Dots Per Inch to use when decompressing XLS pages.

The full list of format server parameters and their usage is in Appendix B, Server Tags for web.xml.

Setting Office 2007 - 2010 Documents to Display Color Output

To display color output in Office 2007 - 2010 documents, set the `xlsxBitDepth` and `docxBitDepth` parameters to 24 and the `xlsxDPI` and `docxDPI` parameters to 200 as shown in the following example:

Example 3.19: Displaying Color Output in Office 2007-2010

```
<init-param>
 <param-name>xlsxDPI</param-name>
 <param-value>200</param-value>
</init-param>
<init-param>
 <param-name>docxBitDepth</param-name>
 <param-value>24</param-value>
</init-param>
<init-param>
 <param-name>docxDPI</param-name>
 <param-value>200</param-value>
</init-param>
<init-param>
 <param-name>xlsxBitDepth</param-name>
 <param-value>24</param-value>
</init-param>
<init-param>
 <param-name>xlsxDPI</param-name>
 <param-value>200</param-value>
</init-param>
<init-param>
 <param-name>docxBitDepth</param-name>
 <param-value>24</param-value>
</init-param>
<init-param>
 <param-name>docxDPI</param-name>
 <param-value>200</param-value>
</init-param>
```

Note:
Aspose.Words.<jdk>.jar, Aspose.Cells.jar and dom4j-1.6.1.jar all need to be on the CLASSPATH for Office 2007 -2010 documents to process without error. Please see *Setting Up Office 2007 - 2010 Support for VirtualViewer* for more information.

Default Configuration Maximizes Performance

Please note that the default configuration for VirtualViewer HTML5 is set to maximize performance. The default settings are the following:

- The bit depth settings for vector formats such as PDF and Word are set to 1. Please note that with the bit depth set at 1 color formats will display as black and white. To view these files in color, set the bit depth to 24.
- The DPI settings for vector formats such as PDF and Word are 200. To increase the quality of an image, set the DPI to a higher value such as 400.
- The default format is set to TIFF_FAX_G4. If you are trying to view another format in color, set the format parameter to the format type.

To improve performance and the speed of loading documents in VirtualViewer Java Content Server, try setting the values of the following parameters in the `web.xml` file as shown below:

Example 3.20: Setting the Parameters in the web.xml File

```
<param-name>documentCacheSize</param-name>
 <param-value>1024000</param-value>
<param-name>wordBitDepth</param-name>
 <param-value>1</param-value>
<param-name>wordDPI</param-name>
```

```
 <param-value>100</param-value>
<param-name>wordFormat</param-name>
 <param-value>JPEG</param-value>
<param-name>pdfBitDepth</param-name>
 <param-value>1</param-value>
<param-name>pdfDPI</param-name>
 <param-value>100</param-value>
<param-name>pdfFormat</param-name>
 <param-value>JPEG</param-value>
<param-name>xlsBitDepth</param-name>
 <param-value>1</param-value>
<param-name>xlsDPI</param-name>
 <param-value>100</param-value>
  <param-value>xlsFormat</param-value>
  <param-value>JPEG</param-value>
```

Note:
Increasing the value of the `documentCacheSize` parameter will improve performance on the client, but will require the server to keep more content in memory and thereby decreasing performance. It is important to find the right balance between the two by performance tuning the cache size during testing.

Caching to Improve Performance

The document cache keeps pages that VirtualViewer has displayed in server memory so that they do not have to be re-rendered the next time they are viewed. This enhances performance but consumes memory on the server. VirtualViewer lets you determine and configure this trade-off between speed vs. memory consumption.

When caching is enabled, the VirtualViewer content server caches the entire document in memory. The HTML5 server caches the pages it receives from the content server. If the content server and HTML5 server are on the same machine (this is common), they will use the same cache.

The caching configuration parameters mentioned below are in the `RetrievalServlet` portion of the content server's web.xml file. These are documented in Appendix B - Servlet Tags for web.xml.

Do you need caching at all?

If your users never view the same page twice, set the `documentCacheSize` parameter to 0 to turn off document caching.

Example 3.21: Turning Off Document Caching

```
<init-param>
```

```
<param-name>documentCacheSize</param-name>
<param-value>0</param-value>
```

Sizing the Cache if You Need It

If your users view the same pages frequently, calculate the number of these pages that should be cached away for faster viewing. This will be **numberOfPagesToCache**.

Next determine how much memory will be used to cache each page, **sizeOfPageInBytes**. To calculate the **sizeOfPageInBytes** value you will need to know:

- Your page size in inches
- Are the pages black & white or color?
  - Black & white uses 1-bit per pixel (bpp).
  - Color uses 24-bpp.
- The desired resolution in dots per inch (DPI). 100 DPI is fine if the document will not be zoomed or printed. A higher DPI may be required if users are zooming in to look at details.

The size can be calculated using the following formula:

(height_in_pixels * width_in_pixels * bits_per_pixel) / (8 bits per byte) = image_size_in_bytes

The value is in bytes and describes the uncompressed size of the page, so it may look rather large. For more information see Recommended JRE Memory Settings.

Here are some examples:

- One black and white 8.5x11 inch page at 100 DPI = (8.5 inches* 300 dpi * 11 inches * 300 dpi) * (1 bit per pixel / 8 bits per byte) = 116875 bytes or 0.1MB per page for **sizeOfPageInBytes**.
- One black and white 8.5x11 inch page at 300 DPI = (8.5 inches* 300 dpi * 11 inches * 300 dpi) * (1 bit per pixel / 8 bits per byte) = 1051875 bytes or roughly 1 MB per page.
- One color 8.5x11 inch page at 300dpi is 25245000 or roughly 25MB per page.

Now, multiply **numberOfPagesToCache** x **sizeOfPageInBytes**. Set the `documentCacheSize` to the calculated cache size to turn on document caching.

If this number is larger than the memory you have available on the system, you can adjust things like the document bit depth, resolution, or number of pages cached.

For example a cache size of 1024000 (1GB) will hold

- 40 full 24-bit color pages at 300 DPI or
- 1,000 black and white pages at 300 DPI or

- 9,187 black and white pages at 100 DPI

Please see Improving Performance or Quality for more information on how to adjust viewing bit depth and resolution.

Defining the Number of Pages Cached in Memory

You can improve performance on the server by allowing VirtualViewer HTML5 to retain pages in the cache. The fewer requests VirtualViewer HTML5 makes to the server the better performance you will see on that server. However, this may impact performance on the client JRE as VirtualViewer HTML5 is now building memory. It is important to find a balance between the two by performance tuning the cache size during testing. The `maxCachePages` parameter will define the number of pages VirtualViewer HTML5 will cache in memory so that it does not always request a page from the server that has already been viewed. The default value is 6.

Example 3.22: Defining the Number of Pages Cached in Memory

```
<init-param>
<param-name>maxCachePages</param-name>
<param-value>6</param-value>
</init-param>
```

Cache Maintenance

If your users modify and save the documents being viewed, set the `clearCacheOnSave` parameter to true (the default) so that older versions of the page are not displayed.

Example 3.23: Clearing Cache on Save

```
<init-param>
<param-name>clearCacheOnSave</param-name>
<param-value>true</param-value>
</init-param>
```

The cache is primarily maintained by your application server. The disk cache will be cleared of all temp files at VirtualViewer startup.

If you start seeing Out of Memory (-1) errors, then you may need to resize your cache as described in the previous section.

Caching and Security

Snowbound Software has no mechanism to selectively remove cached content. However, a Cache Validation interface is provided so you can customize when cached content is permitted to be retrieved. You can implement the `validateCache` method to use your authentication system to validate that the current user is authorized to access the cached page content. See the CacheValidator section in Connecting to Your Document Store in Chapter 4 - Using Advanced Features.

Please see the next topic Chapter 4, Using Advanced Features.

Chapter 4 - Using Advanced Features

This chapter describes how to set up and work with the advanced features in VirtualViewer HTML5.

Virtual Documents

This section describes how to work with virtual documents.

A virtual document is a collection of any combination of documents or pages of documents displayed as a single multi-page document with a single set of thumbnails. The pages can be from documents of different file format types such as AFP, Word, or PDF. The virtual document is viewed and regarded as any normal document would be.

Loading Virtual Documents

To pass a number of documents to the viewer, the value of a `documentId` can start with a special identifier, followed by a string of a comma-separated list of `documentIds`. The list is issued to create the virtual document. The `documentIds` are listed in the order in which the documents are to be compiled for viewing.

Note:
Prior to version 1.8 of VirtualViewer HTML5, exporting virtual documents in original format was not supported for VirtualViewer HTML5. In version 1.8 and above of VirtualViewer HTML5, exporting virtual documents in original format is enabled and virtual documents are exported to a TIFF file using the AJAX Java server. To export a document, select **File > Export Document**.

Virtual Document Syntax

The special identifier is the string `VirtualDocument:` which is then followed by any number of `documentIds`. The syntax can be used any time a normal `documentId` could be used. A `documentId` in the comma-separated list may be specified in the following manner.

Table 4.1: Virtual Document Syntax

File Name	Description
ABC.tif	This specifies that all pages of the document should be included.
ABC.tif[2]	This specifies that only a single page from the document should be included.
ABC.tif[1-3]	This specifies that a range of pages from the document should be included.

Note:
To include non-consecutive pages from a single document, you need to specify the document each time in the virtual document string.

Displaying a Virtual Document

Three documents exist, `ABC.tif`, `EFG.pdf`, and `IJK.doc`, each with three pages. Below are examples of how to create virtual documents.

Example 4.1: Virtual Documents

```
http://localhost:8080/VirtualViewerHTML5/index.html?
documentId=VirtualDocument:ABC.tif,EFG.pdf[2],IJK.doc
```

In the above example, the resultant virtual document would be a 7 page document. Pages 1, 2, and 3 would be all three pages from `ABC.tif`, page 4 would be page 2 from `EFG.pdf`, and pages 5, 6, and 7 would be all three pages from `IJK.doc`.

Example 4.2: Virtual Documents

```
http://localhost:8080/VirtualViewerHTML5/index.html?
documentId=VirtualDocument:ABC.tif[1-2],EFG.pdf,LJK.doc[3]
```

In the above example, the resultant virtual document would be a 6 page document. Pages 1 and 2 would be pages 1 and 2 from `ABC.tif`, page 3, 4, and 5 would be all three pages from `EFG.pdf`, and page 6 would be page 3 from `IJK.doc`.

Printing Virtual Documents

To print a virtual document, select the Print button.

Annotation Security: Watermarks and Redactions

This section describes how to work with annotation security.

The implementation of security for annotations allows each layer to have a permission level assigned to it. This permission level is not inherent in the layer and is only defined when the layer is retrieved by the content handler.

In order to assign a permission level to an annotation layer, the content handler must be implemented or extended and the `getAnnotationProperties` method used.

The Annotation Security Model

The security model is such that when reading annotation layers, various levels of permissions for viewing and working with annotation layers may be specified. The model currently accounts for nine levels on a per layer basis.

Permission levels

Each successive level includes the functionality of previous levels.

If you are storing the annotations as layers (XML files) with a redaction permission level, then you will be able to present them to the users in the viewer as *burned in* but they will not actually be burned into the source document. This would allow you to use an XML tool or create an XML parser that would search and report on these annotation layers (XML files) and give you the information you need to run an offline or server side process such as you described.

Table 4.2: Permission Levels

Permission	Level	Actions Permitted
PERM_HIDDEN	Hidden	The layer is passed to the client but not displayed.
PERM_REDACTION	Redaction	This burns in the annotation layer for viewing.
PERM_PRINT_ WATERMARK	Print Watermark	The user does not see the layer, but it will be burned in for printing.
PERM_VIEW_ WATERMARK	View Watermark	The user may view the layer, but may not hide the layer.
PERM_VIEW	View	The user may view or hide the layer.
PERM_PRINT	Print	The user may also print the layer.
PERM_CREATE	Create	The user may also add an object to the layer.
PERM_EDIT	Edit	The user may also edit an object on the layer, and edit layer properties.
PERM_DELETE	Delete	The user may also delete an object on the layer, and delete the layer.

Level Definitions

Table 4.3: Level Definitions

Permission	Definition
Hidden	If a layer is indicated as having the Hidden permission, the information about the layer will be passed, so that changes done by Page Manipulation will be applied when the annotations are saved. The layer is not displayed to the user even if manipulations are applied.
Redaction	If a layer is indicated as having the Redaction permission, then the servlet will create the working image by applying the layer to the data (i.e. burn in the layer) before passing the working image, so that it becomes part of the image and the data it redacts cannot be seen in any way. The original image is not altered.

Permission	Definition
Print Watermark	If a layer is indicated as having the Print Watermark permission, it shall be passed as a normal layer, but will not be shown to the user. When the document is printed, any layer with Print Watermark permission will be applied to the image before printing.
View Watermark	If a layer is indicated as having the View Watermark permission, it shall be passed as a normal layer. However, the user will not be allowed to show or hide the layer, or manipulate the layer in any way. This layer will never be printed.
View	If a layer is indicated as having the View permission, it shall be passed as a normal layer. The user will be able to hide or show the layer. The user will not be able to add an object, edit an object, delete an object, print the layer, rename the layer, or delete the layer.
Print	If a layer is indicated as having the Print permission, it shall be passed as a normal layer. The user will be able to hide or show the layer, print the layer. The user will not be able to add an object, edit an object, delete an object, or rename or delete the layer.
Create	If a layer is indicated as having the Create permission, it shall be passed as a normal layer. The user will be able to hide or show the layer, print the layer, or add an object to the layer. The user will not be able to edit an object, delete an object, edit the layer properties, or delete the layer.
Edit	If a layer is indicated as having the Edit permission, it shall be passed as a normal layer. The user will be able to hide or show the layer, add an object, edit an object, or edit the layer properties. The user will not be able to delete objects, or delete the layer.
Delete	If a layer is indicated as having the Delete permission, it shall be passed as a normal layer. The user will have full rights to perform any operation on the layer.

Retrieving Annotation Layers

When loading a document, annotation layers will need to be retrieved and have the correct permission level set. The process of loading an annotation layer is as follows:

For each `annotationKey` returned by `getAnnotationNames` the following method will be called.

Example 4.3: Retrieving Annotation Layers

```
public Hashtable getAnnotationProperties (clientInstanceId,
documentKey, annotationKey)
```

This method returns a hash table with the following expected key/value pairs for that annotation layer.

Key/Value Pairs

- The **`permissionLevel`** will determine how the layer is handled. If no value is set, an exception will occur.
- The **`redactionFlag`** determines if the layer has **Mark Layer As Redaction** selected in the client. If no value is set, an exception will occur.

If the `permissionLevel` is set to `PERM_REDACTION`, the value of `redactionFlag` is moot since the client does not receive that layer as an annotation layer.

If `getAnnotationProperties` returns *null*, an exception will occur. This prevents cases where a layer should have strict permissions but for some reason no permission level gets set.

Saving Redaction Layers

If a layer has **Mark Layer As Redaction** selected, when choosing **Save Annotations** the following will occur:

VirtualViewer HTML5 will pass both the `permissionLevel` and the `redactionFlag` to the `saveAnnotationContent` method in a hash table:

Example 4.4: Saving Redaction Layers

```
public void saveAnnotationContent(ContentHandlerInput input)
saveAnnotationContent(ContentHandlerInput input)
(String clientInstanceId, String documentId, String annotationKey, byte
[] data, Hashtable annProperties)
```

Printing Layers

When printing a document, the user may choose to print with or without annotations.

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

A layer which has been given a `permissionLevel` of `PERM_REDACTION` shall always print as part of the image, (since it has been burned into the image), even if the user chose to print without annotations.

Snowbound, FileNet, IDM, and Daeja Annotations

You can save Snowbound , FileNet, IDM, and Daeja annotations. See the sections below for more information on configuring Snowbound, FileNet, IDM, and Daeja annotations.

Configuring Snowbound Annotations

To save annotations in the Snowbound XML format, add the `annotationOutputFormat` parameter with the value set to Snowbound to the servlet web.xml files as shown in the example below:

Example 4.5: Adding the annotationOutputFormat parameter Set to Snowbound

```
<init-param>
 <param-name>annotationOutputFormat</param-name>
<param-value>Snowbound</param-value>
</init-param>
```

Snowbound Annotation Supported Configurations

Server

Table 4.4: Snowbound Annotation Supported Configurations - Server

Parameter Name	Value	File Location
`annotationOutputFormat`	Snowbound	Web.xml

Note:
Snowbound annotations can be used with any configuration of non-required annotation parameters.

Client

None

If VirtualViewer HTML5 is configured to save Snowbound annotations, then any existing annotations that are in the FileNet format are read in as read-only and are not able to be edited or deleted. Edit controls are disabled for annotation layers that are not editable. For example:

- The menu-items for the layer will be visible, but grayed-out in menus such as Select Layer.
- When you right-click an annotation to edit it, the pop-up menu will simply not appear.

Configuring FileNet Annotations

To save annotations in the FileNet XML format, follow the steps below:

1. Add the `annotationOutputFormat` parameter with the value set to FileNet to the servlet web.xml files as shown in the example below:

Example 4.6: Adding the annotationOutputFormat Parameter Set to FileNet

```
<init-param>
 <param-name>annotationOutputFormat</param-name>
<param-value>FileNet</param-value>
</init-param>
```

2. In the config.js file, set the `oneLayerPerAnnotation` parameter to true as shown in the example below:

Example 4.7: Setting oneLayerPerAnnotation to True

```
var oneLayerPerAnnotation = true;
```

FileNet Annotation Supported Configurations

Server

Table 4.5: FileNet Annotation Supported Configurations - Server

Parameter Name	Value	File Location
`annotationOutputFormat`	FileNet	Web.xml

Client

Table 4.6: FileNet Annotation Supported Configurations - Client

Parameter Name	Value	File Location
`base64EncodeAnnotations`	False	Config.js
`oneLayerPerAnnotation`	True	Config.js

Configuring IDM Annotations

To save annotations in the OpenText Integrated Document Management (IDM) Viewer format, add the `annotationOutputFormat` parameter with the value set to IDM to the servlet web.xml files as shown in the example below:

Example 4.8: Adding the annotationOutputFormat parameter Set to IDM

```
<init-param>
<param-name>annotationOutputFormat</param-name>
<param-value>IDM</param-value>
</init-param>
```

IDM Annotation Supported Configurations

Server

Table 4.7: IDM Annotation Supported Configurations - Server

Parameter Name	Value	File Location
`annotationOutputFormat`	IDM	Web.xml

Client

Table 4.8: IDM Annotation Supported Configurations - Client

Parameter Name	Value	File Location
`splitAnnotationLayersByPage`	True	Web.xml
`base64EncodeAnnotations`	False	Config.js

Only the following annotation types are supported when the `annotationOutputFormat` parameter is set to IDM:

- Sticky Note
- Text Edit
- Highlight Rectangle
- Line
- Freehand
- Filled Ellipse(Represented by highlight ellipse)

Note:
IDM annotations are limited to single byte text for VirtualViewer HTML5. Double byte text is not supported for IDM annotations.

Configuring Daeja Annotations

To save annotations in the Daeja format, add the `annotationOutputFormat` parameter with the value set to Daeja to the servlet web.xml files as shown in the example below:

Example 4.9: Adding the annotationOutputFormat parameter Set to Daeja

```
<init-param>
 <param-name>annotationOutputFormat</param-name>
<param-value>Daeja</param-value>
</init-param>
```

Daeja Annotation Supported Configurations

Server

Table 4.9: Daeja Annotation Supported Configurations - Server

Parameter Name	Value	File Location
`annotationOutputFormat`	Daeja	Web.xml

Client

Table 4.10: Daeja Annotation Supported Configurations - Client

Parameter Name	Value	File Location
`base64EncodeAnnotations`	False	Config.js

Annotation Mapping

The table below shows the FileNet annotation and its analogous Snowbound Annotation

Table 4.11: Annotation Mapping

FileNet Annotation	Snowbound Annotation
FileNet Annotation	Snowbound Annotation
Highlight Rectangle	SANN_HIGHLIGHT_RECT
v1-Rectangle	SANN_FILLED_RECT
Arrow	SANN_ARROW
v1-Line	SANN_LINE
v1-Open Polygon	SANN_POLYGON
v1-Highlight Polygon	SANN_FILLED_POLYGON
Pen	SANN_FREEHAND
Stamp	SANN_EDIT
StickyNote	SANN_POSTIT
v1-Oval	SANN_FILLED_ELLIPSE
Text	SANN_EDIT
Transparent Text	SANN_EDIT (Not transparent)
Closed Polygon	SANN_POLYGON
Freehand Line	SANN_FREEHAND

Connecting Your Document Store

VirtualViewer HTML5 comes with a file content handler that connects VirtualViewer HTML5 to your file system. Snowbound Software has content handlers available to connect to web locations (URL content handler), FileNet P8 Enterprise Content Management (ECM), Documentum Webtop ECM and SharePoint. You can create your own custom connector or use Snowbound Professional Services to create a custom content handler for you.

Warning:
VirtualViewer HTML5 includes a sample content handler that connects to the server's File System. This code sample is provided as a starting point to integrate

VirtualViewer HTML5 with your document storage. The content handler sample is not production quality. You should add error checking and permissions checking at the very least before releasing this into production.

What is the Content Handler?

The VirtualViewer HTML5 content handler is a Java class that the servlet will call on to perform various actions concerning the retrieval and storage of content. By default, the VirtualViewer HTML5 servlet uses the sample content handler that Snowbound Software provides, `FileContentHandler`, as its content handler, which merely reads and writes to a file system location. You can find this sample content handler at **VirtualViewerJavaContentServer/WEB-INF/classes/com/snowbound/snapserv/servlet**. It displays files from the `C:/imgs` directory. You are encouraged to use this as a starting point for writing your own custom content handler to integrate VirtualViewer HTML5 into back-end systems. You should create your own content handler to serve up documents from locations that work for your company as well as to add error handling and more robustness for handling requests from multiple users.

How the Content Handler Works

Whenever VirtualViewer HTML5 requests a document, the servlet will first check the cache to see if the document is present. If it is not, it then calls into the content handler for the document. The order of action is as follows:

```
getDocumentContent
getAnnotationNames
getAnnotationContent (once for each layer name returned by
getAnnotationNames)
getBookmarkContent
```

Whenever the user chooses to save the document by choosing **Save Document**, VirtualViewer HTML5 passes the appropriate data to the servlet, which calls the content handler method `saveDocumentComponents`.

Inside `saveDocumentComponents`, the following methods should be called separately when the appropriate data has changed:

```
saveDocumentContent
saveAnnotationContent
saveBookmarkContent
```

Other methods within the content handler are called by various functions in VirtualViewer HTML5.

Defining a Custom Content Handler

The document Content Handler/Connector that VirtualViewer will use is set using the `contentHandlerClass` parameter in the application's web.xml file. Many customers create a custom content handler class that integrates with their document management and security systems. Please see the following example for specifying a custom content handler:

Example 4.10: Setting Up the contentHandlerClass Parameter

```
<init-param>
 <param-name>contentHandlerClass</param-name>
 <param-value>com.snowbound.flexsnap.custom.MyContentHandler
</param-value>
</init-param>
<param-name>contentHandlerClass</param-name>
<param-value>com.mycompany.viewer.DocumentConnector</param-value>
```

VirtualViewer would then look for and invoke your custom content handler at `./WEB-INF/classes/com/mycompany/viewer/DocumentConnector.class`

Snowbound provides two connectors with the VirtualViewer base product; the default File Connector and the URL Connector. Snowbound also provides full featured ICN, FileNet P,8 and Documentum connectors as VirtualViewer options.

To use the default File Connector specify:

Example 4.11: Using the Default File Connector

```
<param-name>contentHandlerClass</param-name>
<param-value>com.snowbound.snapserv.servlet.FileContentHandler</param-
value>

<param name="filePath" value="C:/MyDocuments"/> // folder on the server
<param name='startFile' value='myLogo.gif'> // will display
C:/MyDocuments/myLogo.gif in the client at start up
```

To use the default URL Connector specify:

Example 4.12: Using the Default URL Connector

```
<param-name>contentHandlerClass</param-name>
<param-value>com.snowbound.snapserv.servlet.FileAndURLRetriever</param-
value>

<param-name>baseURL</param-name>
<param-value>http://www.mycompany.com/</param-value>
<param name='startFile' value='myLogo.gif'> // will display
http://www.mycompany.com/myLogo.gif in the client at start up
```

The source code for the default file content handler is provided as a starting point for your own custom connector. The code is located in the VirtualViewer .war file under `\Sample Code\Java Content Handler\SampleContentHandler.java`.

FlexSnapSIContentHandlerInterface

This interface defines methods for retrieving content for VirtualViewer HTML5. Most of the methods take in a single input parameter, which is an instance of the class `ContentHandlerInput`, an extension of `java.util.Hashtable` which contains the data that is required to implement each method.

Likewise, most of the methods return a single value, which is an instance of the class `ContentHandlerResult`, also and an extension of `java.util.Hashtable` which contains the data required to complete the method.

CacheValidator

This interface defines a method that will be called when a document is requested that is in the cache to determine whether or not the cache may be used to retrieve the document or the normal content handler sequence must be called.

The document cache speeds up access to documents by saving the rendering the first time a document is viewed. When it is viewed for the second time, the rendering can be fetched from the document cache and re-used.

When multiple users are viewing documents, documents that should be secured may end up in the document cache. To prevent a user that does not have permission from viewing a high security document, use the cache validator to check the user’s permission before allowing a document to be fetched from the cache for that user.

The cache validator can also be used to prevent high security documents from being stored in the cache.

To use this feature, your custom content handler must implement `com.snowbound.snapserv.servlet.CacheValidator` in addition to `FlexSnapSIContentHandlerInterface`.

CacheValidator Method Detail

`validateCache`

```
public ContentHandlerResult validateCache (ContentHandlerInput
input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Determines whether or not the specified cache put or get is allowed.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_ANNOTATION_ID"`		Either `ContentHandlerInput.VALUE_CACHE_GET` or `ContentHandlerInput.VALUE_CACHE_PUT`.

Returns

A `ContentHandlerResult` object with the following key/value pairs:
`ContentHandlerResult.KEY_USE_OF_CACHE_ALLOWED` - either
`Boolean.FALSE or Boolean.TRUE.`

Event Notification and Handling

`eventNotification`

The VirtualViewer HTML5 client sends event notifications to the VirtualViewer HTML5's content handler on the server whenever the user does something that triggers an audited event. Event triggers include opening a document or going to another page in the document.

```
public ContentHandlerResult eventNotification (ContentHandlerInput
input) throws FlexSnapSIAPIException
```

Implement this content handler method to receive event notifications.

This allows event notification when the `enableEventNotifications` applet parameter is set to true as shown in the example below. The `eventNotification` method is called whenever a document is successfully retrieved from the content server's internal cache. The default value is false for event notification to be turned off. Please see Appendix A - Applet Parameters in the *VirtualViewer Client Administrator's Guide* for more information on setting up the `enableEventNotifications` applet parameter.

```
<param name="enableEventNotifications" value="true">
```

The following auditing events trigger event notification:

Page request,
Save annotation,
Save document,
Print,
Export,
Document close

The default file content handler logs these events to the web server's log. For example, these events might appear in a TomCat web server log:

```
[exec] 04-30-2012 16:00:13 FileContentHandler.eventNotification
[exec] 04-30-2012 16:00:13 Key: KEY_EVENT_PAGE_REQUESTED_NUMBER,
value: 0
[exec] 04-30-2012 16:00:13 Key: KEY_DOCUMENT_ID, value: 6-Pages.tif
[exec] 04-30-2012 16:00:13 Key: cacheBuster, value:
0.7112829455936928
[exec] 04-30-2012 16:00:13 Key: KEY_EVENT, value: VALUE_EVENT_PAGE_
REQUESTED
[exec] 04-30-2012 16:00:13 FileContentHandler.eventNotification
[exec] 04-30-2012 16:00:13 Key: KEY_EVENT_PAGE_REQUESTED_NUMBER,
value: 0
[exec] 04-30-2012 16:00:13 Key: KEY_DOCUMENT_ID, value: 6-Pages.tif
[exec] 04-30-2012 16:00:13 Key: cacheBuster, value:
0.1953655708607337
[exec] 04-30-2012 16:00:13 Key: KEY_EVENT, value: VALUE_EVENT_PAGE_
REQUESTED
```

The server `loglevel` must be set to **Info** in order to see the event notifications. The `logLevel` must be set to **Finest** to see all the Key event details in the log file.

You can change the information being logged and the required `loglevel` by modifying the `eventNotification` method in the sample FileContentHandler.

The `eventNotification` method in the content handler can be customized to meet your needs. For example, you could add code to send a message to an audit logging system for certain events or change the log messages to match your company's standard format.

The following are the details for each event:

Parameters

The ContentHandlerInput hash table will contain a variety of elements depending on the type of event being logged, all values are strings:

Key	Type	Description
`"KEY_EVENT"`	`String`	One of the VALUE_EVENT_* values
`"VALUE_EVENT_PAGE_REQUESTED"`	`String`	The event being logged is a page request.
`"KEY_EVENT_PAGE_REQUESTED_NUMBER"`	`String`	The page number requested (zero-based).
`"VALUE_EVENT_SAVE_ANNOTATION"`	`String`	The event being logged is a save annotation request.
`"KEY_EVENT_SAVE_ANNOTATION_LAYER_NAME_BASE*"`	`String`	The base name of the keys containing the layer names. There will be one of these for each layer. For example: "KEY_EVENT_SAVE_ANNOTATION_LAYER_NAME_BASE0"
`"VALUE_EVENT_PRINT"`	`String`	The event being logged is a print request.
`"KEY_EVENT_PRINT_PAGE_NUMBERS"`	`String`	The page range being printed, in the format '0-4'

Key	Type	Description
`"VALUE_EVENT_EXPORT"`	`String`	The event being logged is a document export request.
`"KEY_ANNOTATION_ID"`	`String`	The name of the annotation layer.

Key/Value pairs passed on page request:

```
Key = ContentHandlerResult.KEY_EVENT
Value = ContentHandlerResult.VALUE_EVENT_PAGE_REQUESTED

Key = ContentHandlerResult.KEY_DOCUMENT_ID
Value = <documentId>

Key = ContentHandlerResult.KEY_CLIENT_INSTANCE_ID
Value = <clientInstanceId>

Key = ContentHandlerResult.KEY_EVENT_PAGE_REQUESTED
Value = <page number>
```

Key/Value pairs passed on annotation save:

```
Key = ContentHandlerResult.KEY_EVENT
Value = ContentHandlerResult.VALUE_EVENT_SAVE_ANNOTATION

Key = ContentHandlerResult.KEY_DOCUMENT_ID
Value = <documentId>

Key = ContentHandlerResult.KEY_CLIENT_INSTANCE_ID
Value = <clientInstanceId>
```

Key/Value pairs passed on print:

```
Key = ContentHandlerResult.KEY_EVENT
Value = ContentHandlerResult.VALUE_EVENT_PRINT

Key = ContentHandlerResult.KEY_DOCUMENT_ID
Value = <documentId>

Key = ContentHandlerResult.KEY_CLIENT_INSTANCE_ID
Value = <clientInstanceId>

Key = ContentHandlerResult.KEY_EVENT_PAGE_REQUESTED
Value = <page number>
```

Key/Value pairs passed on export:

```
Key = ContentHandlerResult.KEY_EVENT
Value = ContentHandlerResult.VALUE_EVENT_EXPORT

Key = ContentHandlerResult.KEY_DOCUMENT_ID
Value = <documentId>

Key = ContentHandlerResult.KEY_CLIENT_INSTANCE_ID
Value = <clientInstanceId>

Key = ContentHandlerResult.KEY_EVENT_EXPORT_FORMAT_NAME
Value = <format>
```

Key/Value pairs passed on document close:

```
Key = ContentHandlerResult.KEY_EVENT
Value = ContentHandlerResult.VALUE_EVENT_CLOSE_DOCUMENT

Key = ContentHandlerResult.KEY_DOCUMENT_ID
Value = <documentId>

Key = ContentHandlerResult.KEY_CLIENT_INSTANCE_ID
Value = <clientInstanceId>
```

Key/Value pairs passed when retrieved from the internal cache:

```
Key = ContentHandlerResult.KEY_EVENT
Value = ContentHandlerResult.VALUE_EVENT_DOCUMENT_RETRIEVED_FROM_
CACHE

Key = ContentHandlerResult.KEY_DOCUMENT_ID
Value = <documentId>

Key = ContentHandlerResult.KEY_CLIENT_INSTANCE_ID
Value = <clientInstanceId>
```

Use the following sample `eventNotification` method in a custom content handler to make use of the new event for cache retrieval:

Example 4.13: eventNotification Method for New Event for Cache Retrievel

```
public ContentHandlerResult eventNotification
   (ContentHandlerInput input)
 throws FlexSnapSIAPIException
String eventType = (String) input.get("KEY_EVENT");
```

```
if (eventType.equals("VALUE_EVENT_DOCUMENT_RETRIEVED_FROM_
CACHE"))
{
String documentId = input.getDocumentId();
System.out.println("Document retrieved from cache: " +
documentId);
// do getDocumentContent related tasks here because
getDocumentContent will not be called
}
return ContentHandlerResult.VOID;
}
```

Returns

A `ContentHandlerResult` object or null. The return value is currently ignored.

Extracting Parameters from ContentHandlerInput

There are two methods with which you can extract parameters from the `ContentHandlerInput` hash table.

The first method is by using predefined functions. For example, the method

`getDocumentContent(ContentHandlerInput input)`

typically contains two parameters, `clientInstanceId` and `documentId`. In order to extract each parameter, you would do the following:

```
String clientID = input.getClientInstanceId();
String documentID = input.getDocumentId();
```

Below is a table with the existing methods for extracting parameter data.

Table 4.12: Method Summary

Method	Description
`getAnnotationContent()`	Returns a byte array containing the content of a specified annotation layer.
`getAnnotationId()`	Returns the `annotationId` parameter.
`getAnnotationLayers()`	Returns an array of annotation layers.
`getAnnotationProperties()`	Returns a hash table containing Annotation Properties for a given layer.
`getBookmarkContent()`	Returns a byte array containing the specified bookmark XML content.
`getClientInstanceId()`	Returns the `clientInstanceId` parameter.
`getClientPreferencesXML()`	Returns an XML string for the specified client preferences.
`getDocumentContent()`	Returns a byte array containing the specified document

Method	Description
	content.
`getDocumentFile()`	Returns the `getdocumentFile` parameter.
`getDocumentId()`	Returns `documentId` parameter.
`getHttpServletRequest()`	Returns a `HttpServletRequest` object.

The second method is by explicitly calling the get function on the input hash table. For example, to retrieve the same values as the previous example, you would do the following:

```
input.get(ContentHandlerInput.KEY_CLIENT_INSTANCE_ID);
input.get(ContentHandlerInput.KEY_DOCUMENT_ID);
```

Below is a table with the existing keys for the hash table for extracting parameter data.

Table 4.13: Existing Keys for the Hash Table to Extract Parameter Data

Property	Type	Description
`"KEY_ANNOTATION_CONTENT"`	`byte[]`	The annotation data for a given layer.
`"KEY_ANNOTATION_ID"`	`String`	The name of the annotation layer.
`KEY_ANNOTATION_LAYERS"`	`AnnotationLayer[]`	The information for all annotation layers.
`"KEY_ANNOTATION_PROPERTIES`	`Hashtable`	The properties for an annotation layer.
`"KEY_BOOKMARK_CONTENT"`	`byte[]`	The XML data for bookmarks.
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_CLIENT_PREFERENCES_XML"`	`String`	The XML data for client preferences.
`"KEY_DOCUMENT_CONTENT"`	`byte[]`	The data of the document.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_HTTP_SERVLET_REQUEST"`	`HttpServletRequest`	The Java `HttpServletRequest` object.
`"KEY_MERGE_ANNOTATIONS"`	`boolean`	Indicates if annotations were burned in or not.

Populating Parameters for ContentHandlerInput

Occasionally, the `ContentHandlerInput` hash table may need to have parameters manually added. This may be done using the `ContentHandlerInput.put(key, value)` using the desired key listed below, or by creating your own key.

```
input.put(ContentHandlerInput.KEY_DOCUMENT_ID, test.pdf);
```

Populating Parameters for ContentHandlerResult

Return values for each method are handled in a similar fashion to `ContentHandlerInput`. Most of the methods have a return class of `ContentHandlerResult`, which is an extension of `java.util.Hashtable`.

The required data for each method should be put into the `ContentHandlerResult` return object with the `ContentHandlerResult.put(key, value)`, using the key/value pairs specified in the method's documentation.

```
result.put(ContentHandlerResult.DOCUMENT_ID_TO_RELOAD, test_b.pdf);
```

Table 4.14: Property Descriptions

Property	Description
`"DOCUMENT_ID_TO_RELOAD"`	The `documentId` to load after a save is made.
`"ERROR_MESSAGE"`	The error message if there is an error.
`"KEY_ANNOTATION_CONTENT"`	The annotation data for a given layer.
`"KEY_ANNOTATION_NAMES"`	The names of all annotation layers.
`"KEY_ANNOTATION_PROPERTIES"`	The properties for a given annotation layer.
`"KEY_AVAILABLE_DOCUMENT_IDS"`	The array of `documentId`'s for `availableDocument` mode.
`"KEY_BOOKMARK_CONTENT"`	The XML data for bookmarks.
`"KEY_CLIENT_PREFERENCES_XML"`	The XML data for client preferences.
`"KEY_DOCUMENT_CONTENT"`	The data of the document.
`"VOID"`	Used for null or void returns.

How to Return an Error for Display in the Client

There are two ways to return error messages to the client. The method that works with all operations is to throw a `FlexSnapSIAPIException`. For example:

```
if (currentSecLevel.equals("0")) {
 throw new FlexSnapSIAPIException("Security violation detected");
```

For **Send** and **Save** operations you may return an error message through `ContentHandlerResult.ERROR_MESSAGE` as shown in the following example:

```
if (currentSecLevel.equals("0")) {

ContentHandlerResult failResult = new ContentHandlerResult();
failResult.put(ContentHandlerResult.ERROR_MESSAGE, "Security violation
detected");
failResult.put(ContentHandlerResult.KEY_DOCUMENT_DISPLAY_NAME, "Security
error");
return failResult;

}
```

Content Handler Methods

Below is a table that lists the methods within the content handler broken into two groups corresponding with the two classes `FlexSnapSIContentHandlerInterface` and `FlexSnapSISaverInterface`. The following section defines each method in more detail.

Table 4.15: FlexSnapSIContentHandlerInterface

Return Value	Method
ContentHandlerResult	`deleteAnnotation(ContentHandlerInput input)` Called when the client has requested to delete the specified annotation layer.
ContentHandlerResult	`eventNotification(ContentHandlerInput input)` Implement this content handler method to receive event notifications.
ContentHandlerResult	`getAnnotationContent(ContentHandlerInput input)` Returns the content for the specified annotation key in the form of a byte array.
ContentHandlerResult	`getAnnotationNames(ContentHandlerInput input)` Returns an array of annotation object names for the specified `clientInstance` and `documentKey` array.
ContentHandlerResult	`getAnnotationProperties(ContentHandlerInput input)` Returns the properties for a specified annotation key (layer) in the form of a hash table.
ContentHandlerResult	`getAvailableDocumentIds(ContentHandlerInput input)` Returns an array containing the set of `documentIds` available for viewing for the specified `clientInstance`.
ContentHandlerResult	`getBookmarkContent(ContentHandlerInput input)` Returns the bookmark XML content for the specified `documentKey` in the form of a byte array.
ContentHandlerResult	`getClientPreferencesXML(ContentHandlerInput input)` Retreives an XML String containing the preferences for the specified `clientInstanceId`.
ContentHandlerResult	`getDocumentContent(ContentHandlerInput input)` Returns the content for the specified content key in the form of a byte array.
boolean	`hasAnnotations(ContentHandlerInput input)` Returns true if there is annotation content associated with the specified document.
void	`init(javax.servlet.ServletConfig config)` Performs any necessary configuration tasks.
ContentHandlerResult	`saveClientPreferencesXML(ContentHandlerInput input)` Saves an XML String containing the preferences for the specified `clientInstanceId`.

Return Value	Method
ContentHandlerResult	`sendDocumentContent(ContentHandlerInput input)` This method gets called to send an image via a mechanism defined by the implementor.

Table 4.16: FlexSnapSISaverInterface

`FlexSnapSIContentHandlerInterface` **extends FlexSnapSISaverInterface**	
ContentHandlerResult	`publishDocument (ContentHandlerInput input)`
ContentHandlerResult	`saveAnnotationContent (ContentHandlerInput input)`
ContentHandlerResult	`saveBookmarkContent (ContentHandlerInput input)`
ContentHandlerResult	`saveDocumentComponents (ContentHandlerInput input)`
ContentHandlerResult	`saveDocumentComponentsAs (ContentHandlerInput input)`
ContentHandlerResult	`saveDocumentContent (ContentHandlerInput input)`

Table 4.17: Cache Validator

Return Value	Method
ContentHandlerResult	`validateCache(ContentHandlerInput input)` Determines whether or not the specified cache put or get is allowed..

VirtualViewerContentHandlerInterface Method Detail

deleteAnnotation

```
public ContentHandlerResult deleteAnnotation (ContentHandlerInput
input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called when the client has requested to delete the specified annotation layer.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_ANNOTATION_ID"`	`String`	The name of the annotation layer.

Returns

A `ContentHandlerResult` object or null. The return value is currently ignored.

`getAnnotationContent`

```
public ContentHandlerResult getAnnotationContent
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request the content for the specified annotation key in the form of a byte array. This method returns all the content of the annotation files (layers) that were called by `getAnnotationNames`.

Example 4.14: Specifying the getAnnotationContent Content Handler Method

```
public ContentHandlerResult getAnnotationContent
    (ContentHandlerInput input)
    throws FlexSnapSIAPIException
{
  String clientInstanceId = input.getClientInstanceId();
  String documentKey = input.getDocumentId();
  String annotationKey = input.getAnnotationId();
  ContentHandlerResult result = new ContentHandlerResult();
// Code to retrieve annotation file goes here
  try
  result.put(ContentHandlerResult.KEY_ANNOTATION_CONTENT, annData);
  return result;
}
```

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_ANNOTATION_ID"`	`String`	The name of the annotation layer.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
"KEY_ANNOTATION_CONTENT"	byte[]	The annotation data for a given layer.
"KEY_ANNOTATION_DISPLAY_NAME"	String	The display name of the annotation layer.

getAnnotationNames

```
public ContentHandlerResult getAnnotationNames (ContentHandlerInput
input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request an array of annotation object names for the specified `clientInstance` and `documentKey` array. This method is called to retrieve all the annotation object names for the specified document that was requested by `getDocumentContent`.

Example 4.15: Specifying the getAnnotationNames Content Handler Method

```
public ContentHandlerResult getAnnotationNames(ContentHandlerInput
input)
    throws FlexSnapSIAPIException
{
  String clientInstanceId = input.getClientInstanceId();
  String documentKey = input.getDocumentId();
  String[] arrayNames = new String[2];
  arrayNames[0] = "layerOne";
  arrayNames[1] = "layerTwo";
  ContentHandlerResult result = new ContentHandlerResult();
  result.put(ContentHandlerResult.KEY_ANNOTATION_NAMES,
     arrayNames);
  return result;
}
```

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
"KEY_CLIENT_INSTANCE_ID"	String	Value of the `clientInstanceId` parameter.
"KEY_DOCUMENT_ID"	String	The name or ID of the document.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
"KEY_ANNOTATION_NAMES"	String	The names of all annotation layers.

getAnnotationProperties

```
public ContentHandlerResult getAnnotationProperties
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request the properties for a specified annotation layer in the form of a hash table. This method is used to request the permission level of the specified annotation layer. Here you will give each annotation layer retrieved or saved via the `getAnnotationContent` or `saveAnnotationContent` a permission level. Permissions are defined in Annotation Security, but they will allow you to set permissions 1-9 for each layer. This layer is then passed to the viewer with those permissions set and will restrict functionality the user has based on the permission set.

Example 4.16: getAnnotationProperties

```
public ContentHandlerResult getAnnotationProperties(ContentHandlerInput
input)
throws FlexSnapSIAPIException
{
Logger.getInstance().log(Logger.FINEST, " Begin getAnnotationProperties
method...");
String clientID = input.getClientInstanceId();
String documentKey = input.getDocumentId();
String annotationKey = input.getAnnotationId();
Logger.getInstance().log(Logger.FINEST, " getAnnotationProperties,
CLIENT ID: " + clientID);
Logger.getInstance().log(Logger.FINEST, " getAnnotationProperties, DOC
KEY: " + documentKey);
Logger.getInstance().log(Logger.FINEST, " getAnnotationProperties, ANN
KEY: " + annotationKey);
Hashtable properties = new Hashtable();

String baseAnnFilename = documentKey + "." + annotationKey;
String annFilename = gFilePath + baseAnnFilename + ".ann";
String redactionEditFilename = gFilePath + baseAnnFilename + "-
redactionEdit.ann";
String redactionBurnFilename = gFilePath + baseAnnFilename + "-
redactionBurn.ann";

// Is it a regular annotation layer?
File file = new File(annFilename);

if (file.exists() == true)
{
properties.put("permissionLevel", PERM_DELETE);
properties.put("redactionFlag", new Boolean(false));
Logger.getInstance().log(Logger.FINEST, " Permission: DELETE, false");
}

// Is it a redaction layer the user can edit?
file = new File(redactionEditFilename);
if (file.exists() == true)
{
properties.put("permissionLevel", PERM_DELETE);
properties.put("redactionFlag", new Boolean(true));
Logger.getInstance().log(Logger.FINEST, " Permission: DELETE, true");
}

// Is it a redaction layer the user can NOT edit?
file = new File(redactionBurnFilename);

if (file.exists() == true)
{
properties.put("permissionLevel", PERM_REDACTION);
properties.put("redactionFlag", new Boolean(true));
Logger.getInstance().log(Logger.FINEST, " Permission: REDACT, true");
}
ContentHandlerResult result = new ContentHandlerResult();
result.put(ContentHandlerResult.KEY_ANNOTATION_PROPERTIES, properties);
Logger.getInstance().log(Logger.FINEST, " End of
getAnnotationProperties method...");
return result;
}
```

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_ANNOTATION_ID"`	`String`	The name of the annotation layer.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"KEY_ANNOTATION_PROPERTIES"`	`Hashtable`	The properties for a given annotation layer.

`getAvailableDocumentIds`

```
public ContentHandlerResult getAvailableDocumentIds
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request an array containing the set of `documentIds` available for viewing for the specified `clientInstance`.

Example 4.17: Specifying the getAvailableDocumentIds Content Handler Method

```
public ContentHandlerResult getAvailableDocumentIds
  (ContentHandlerInput input)
{
  String clientInstanceId = input.getClientInstanceId();
  File imgDirectory = new File(gFilePath);
  String[] myArray = imgDirectory.list(this);
  ContentHandlerResult result = new ContentHandlerResult();
result.put(ContentHandlerResult.KEY_AVAILABLE_DOCUMENT_IDS,
  myArray);
  return result;
}
```

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"KEY_AVAILABLE_DOCUMENT_IDS"`	`String[]`	The `documentId's` for `availableDocument` mode.

`getBookmarkContent`

```
public ContentHandlerResult getBookmarkContent (ContentHandlerInput
input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request the `bookmark XML` content for the specified `documentId` in the form of a string. For example, The `FileRetriever` class treats the `documentId` as a file name, and returns the corresponding contents in the byte array.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"KEY_BOOKMARK_CONTENT"`	`byte[]`	The XML data for bookmarks.

getClientPreferencesXML

```
public ContentHandlerResult getClientPreferencesXML
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to retrieve an XML String containing the preferences for the specified `clientInstanceId`.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_PREFERENCES_XML"`	`String`	The XML data containing the preferences for the specified client.

getDocumentContent

```
public ContentHandlerResult getDocumentContent (ContentHandlerInput
input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request the content for the specified content key in the form of a byte array. For example, The `FileRetriever` class treats the `documentId` as a file name, and returns the corresponding contents in the byte array. This method is called to request the document from your file system in the form of a byte array. It uses `documentId` to identify the document being requested.

Example 4.18: Specifying the getDocumentContent Content Handler Method

```
public ContentHandlerResult getDocumentContent
  throws FlexSnapSIAPIException
{
  String clientInstanceId = input.getClientInstanceId();
  String key = input.getDocumentId();
  String fullFilePath = gFilePath + URLDecoder.decode(key);
  File file = new File(fullFilePath);
  ContentHandlerResult result = new ContentHandlerResult();
  result.put(ContentHandlerResult.KEY_DOCUMENT_CONTENT, getFileBytes
(file));
  return result;
}
```

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
"KEY_HTTP_SERVLET_ REQUEST"		The standard `HttpServletRequest` data.
"KEY_CLIENT_INSTANCE_ID"	String	Value of the `clientInstanceId` parameter.
"KEY_DOCUMENT_ID"	byte[]	The contents of the document.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
"KEY_DOCUMENT_CONTENT"	byte[]	The contents of the document.

hasAnnotations

```
public boolean hasAnnotations(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Returns true if there is annotation content associated with the specified document.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
"KEY_CLIENT_INSTANCE_ID"	String	Value of the `clientInstanceId` parameter.
"KEY_DOCUMENT_ID"	String	The contents of the document.

Returns

True, if there is annotation content associated with the specified document.

init

```
public void init(javax.servlet.ServletConfig config)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Performs any necessary configuration tasks.

Parameters

`config` -The ServletConfig object for the FlexSnap: SI Servlet.

Returns

void

`saveClientPreferencesXML`

```
public ContentHandlerResult saveClientPreferencesXML
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to save an XML String containing the preferences for the specified `clientInstanceId`.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_CLIENT_PREFERENCES_XML"`	`String`	The XML data for client preferences.

Returns

A `ContentHandlerResult` object or null. The return value is currently ignored.

`sendDocumentContent`

```
public ContentHandlerResult sendDocumentContent
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

This method gets called when Send Document or Send Document With Annotations is chosen. While this method is often implemented to send the image via email, it is not a given.

The following example will use the `sendDocumentContent` method to overwrite the original file rather than create a new one:

Example 4.19: sendDocumentContent Method: Overwrite Original File

```
/**
* @see
com.snowbound.snapserv.servlet.FlexSnapSIContentHandlerInterface#sendDo-
cumentContent
(ContentHandlerInput)
*/
public ContentHandlerResult sendDocumentContent(ContentHandlerInput
input)
throws FlexSnapSIAPIException
{
 ContentHandlerResult retVal = new ContentHandlerResult();
 HttpServletRequest request = input.getHttpServletRequest();
 String clientInstanceId = input.getClientInstanceId();
 String documentKey = input.getDocumentId();
 boolean mergeAnnotations = input.mergeAnnotations();
 byte[] data = input.getDocumentContent();
 File saveFile = new File(gFilePath + documentKey);
 ClientServerIO.saveFileBytes(data, saveFile);
 return retVal;
}
```

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
"KEY_HTTP_SERVLET_REQUEST"		The standard Java `HttpServletRequest` object.
"KEY_CLIENT_INSTANCE_ID"	String	Value of the `clientInstanceId` parameter.
"KEY_DOCUMENT_ID"	String	The name or ID of the document.
"KEY_DOCUMENT_FORMAT"	Integer	An Integer value indicating the document's format. For more information, see Appendix D, Supported File Formats.
"KEY_MERGE_ANNOTATIONS"	Boolean	The name of the annotation layer.
"KEY_DOCUMENT_CONTENT"	byte[]	The data of the document.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
"DOCUMENT_ID_TO_RELOAD"	String	The `documentId` to load after a save is made.

If a value is set for `DOCUMENT_ID_TO_RELOAD`, then VirtualViewer HTML5will load or reload the specified document when the publish has been completed. If no value for `DOCUMENT_ID_`

`TO_RELOAD` is set, then the default behavior is for the current page to remain loaded in the viewer.

VirtualViewerSaverInterface Method Detail

`publishDocument`

```
public ContentHandlerResult publishDocument (ContentHandlerInput
input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to publish a document with annotations to PDF. This method is invoked by the **File > Publish Document** command.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_HTTP_SERVLET_REQUEST"`		The standard Java `HttpServletRequest` object.
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_DOCUMENT_FORMAT"`	`Integer`	An Integer value indicating the document's format. For more information, see Appendix D, Supported File Formats.
`"KEY_DOCUMENT_CONTENT"`	`byte[]`	The data of the document.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"DOCUMENT_ID_TO_RELOAD"`	`String`	The `documentId` to load after a save is made.

If a value is set for `DOCUMENT_ID_TO_RELOAD`, then VirtualViewer HTML5will load or reload the specified document when the publish has been completed. If no value for `DOCUMENT_ID_TO_RELOAD` is set, then the default behavior is for the current page to remain loaded in the viewer.

`saveAnnotationContent`

```
public ContentHandlerResult saveAnnotationContent
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

This method is called to save each annotation layer changed or created in the viewer.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_HTTP_SERVLET_ REQUEST"`		The standard Java `HttpServletRequest` data.
`KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_ANNOTATION_ID"`	`String`	The name or ID of the annotation layer.
`"KEY_ANNOTATION_CONTENT"`	`byte[]`	The annotation data of the document.

Returns

A `ContentHandlerResult` object or null. The return value is currently ignored.

`saveBookmarkContent`

```
public ContentHandlerResult saveBookmarkContent
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to save bookmark data. This method is called to save the XML components of the bookmarks created.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_HTTP_SERVLET_ REQUEST"`		The standard Java `HttpServletRequest` object.
`"KEY_CLIENT_INSTANCE_ ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_BOOKMARK_CONTENT"`	`byte[]`	The XML data for bookmarks.

Returns

A `ContentHandlerResult` object or null. The return value is currently ignored.

`saveDocumentComponents`

```
public ContentHandlerResult saveDocumentComponents
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to save all components of a document including the document, annotations, and bookmarks. This method is invoked by **File > Save Document** in the VirtualViewer HTML5.

Within this method the individual methods `saveDocumentContent`, `saveAnnotationContent` and `saveBookmarkContent` are each typically called to handle saving of each type of content separately.

Calling one of those methods alone can cause issues, such as if you have deleted a page and only called `saveDocumentContent`, the annotations will have an extra page if you do not also save the annotations.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_HTTP_SERVLET_REQUEST"`		The standard Java `HttpServletRequest` object.
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_DOCUMENT_CONTENT"`	`byte[]`	The data of the document.
`"KEY_DOCUMENT_FORMAT"`	`Integer`	An Integer value indicating the document's format. For more information, see Appendix D, Supported File Formats.
`"KEY_ANNOTATION_LAYERS"`	`AnnotationLayer[]`	The information for all annotation layers.
`"KEY_BOOKMARK_CONTENT"`	`byte[]`	The XML data for bookmarks.

Using `KEY_ANNOTATION_LAYERS`

In order to save each annotation layer, `saveAnnotationContent` must be called once for each existing layer that has been changed or created. `KEY_ANNOTATION_LAYERS` is an object that contains all the information for all annotation layers of a given document that have changed or been created. In order to retrieve the information for each individual layer, there are three methods you can call on the `AnnotationLayer`[] object.

Once you have set the proper information in the `ContentHandlerInput` object, you can call `saveAnnotationContent`.

Example 4.20: Calling saveAnnotationContent

```
AnnotationLayer[] ann = input.getAnnotationLayers();
for (int annIndex = 0; annIndex < annotations.length; annIndex++)
{
input.put(ContentHandlerInput.KEY_CLIENT_INSTANCE_ID,
clientInstanceId);
input.put(ContentHandlerInput.KEY_DOCUMENT_ID, documentId);
input.put(ContentHandlerInput.KEY_ANNOTATION_ID, ann
[index].getLayerName());
input.put(ContentHandlerInput.KEY_ANNOTATION_CONTENT, ann
[index].getData());
input.put(ContentHandlerInput.KEY_ANNOTATION_PROPERTIES,
  ann[index].getProperties());
saveAnnotationContent(input);
}
```

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"DOCUMENT_ID_TO_RELOAD"`	`String`	The `documentId` to load after a save is made.

Select the **Annotations > Select Layer** menu and click on the annotation layer name that you want to edit. When you are trying to change an annotation object on a specific layer, you need to be on that layer. Make sure that the check mark appears next to that annotation layer name, then try editing the object.

`saveDocumentComponentsAs`

```
public ContentHandlerResult saveDocumentComponentsAs
(ContentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called as an alternative to `saveDocumentComponents`. This method is typically used to create alternate copies of a document as new content, rather than create new versions or renditions of the original content.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_HTTP_SERVLET_ REQUEST"`		The standard Java `HttpServletRequest` object.
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_DOCUMENT_CONTENT"`	`byte[]`	The data of the document.
`"KEY_ANNOTATION_LAYERS"`	`AnnotationLay-er[]`	The information for all annotation layers.
`"KEY_BOOKMARK_CONTENT"`	`byte[]`	The XML data for bookmarks.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"DOCUMENT_ID_TO_RELOAD"`	`String`	The `documentId` to load after a save is made.

`saveDocumentContent`

```
public ContentHandlerResult saveDocumentContent(ContentHandlerInput
input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to save the content of a document.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_HTTP_SERVLET_ REQUEST"`		The standard Java `HttpServletRequest` data.
`"KEY_CLIENT_INSTANCE_ID"`	`String`	Value of the `clientInstanceId` parameter.
`"KEY_DOCUMENT_ID"`	`String`	The name or ID of the document.
`"KEY_DOCUMENT_CONTENT"`	`byte[]`	The data of the document.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
`"DOCUMENT_ID_TO_RELOAD"`	`String`	The `documentId` to load after a save is made.

Please see the next topic Appendix A, Config.js Parameters.

Appendix A - Config.js Parameters

Descriptions of Config.js Parameters

This table lists and describes the supported config.js parameters.

Warning:
Please make a backup copy of the `config.js` file before you edit it.

Table A.1: Config.js Parameters

Name	Default	Definition
`servletPath`	`"/VirtualViewerHTML5"`	Specifies the path to the servlet for the .NET configuration.
`maxZoomPercent`	`1000`	Sets the percentage to stop allowing users to zoom the image.
`zoomTimeout`	`500`	Sets the wait in X milliseconds before requesting the zoomed image. This variable spares the server load by only requesting the final zoom level when the user selects zoom several times quickly.
`pictureControlsTimeout`	`200`	Sets the wait in X milliseconds before requesting the adjusted images. This exists to throttle the number of image requests sent to the server. If the user quickly slides the sliders back and forth this will wait X seconds before requsting the updated image.
`waitDialogTimeout`	`1000`	Sets the wait in X milliseconds before displaying the "Please wait while your image is loaded." dialog message.
`polygonNubSize`	`10`	Sets the size of the "handle" used to resize annotations.
`polygonNubFillColor`	`"rgba(0,0,255,.40)"`	Sets the color of the "handle" used to resize annotations and

Name	Default	Definition
		to indicate the "end zone."
`imageScrollBars`	`true`	If set to true, turns on the scroll bars for the image display. This disables the pan tool. To turn on the pan tool, set the value to false. Please see the release notes for any updates.
`sendDocumentWithAnnotations`	`false`	If set to true, includes annotations when `sendDocument` is called.
`retainViewOptionsBetweenPages`	`true`	Maintains the same zoom, rotation, fit, flip and other settings when switching between pages. If set to true, the zoom level will be reset to the `defaultZoomMode` setting when switching between pages. If set to false, `retainViewOptionsBetweenPages` should ensure that the viewer does not go back to the `defaultZoomMode` setting when switching between pages.
`defaultZoomMode`	`vvDefines.zoomModes.fitWindow`	Sets the default zoom mode. You can use any of the following variables: `fitWidth` - Fits the page to the width of the image panel. `fitHeight` - Fits the page to the height of the image panel. `fitPanel` - Fits the page in the panel, regardless of landscape or portrait. `fitImage` - Fits the page to 100 percent.

Name	Default	Definition
		`fitLast` - Fits to the last zoom level of the last viewed page. If this value is set for this parameter, then the initial zoom level for the first page of the first document viewed is `fitWindow`. To retain the `fitLast` zoom level between documents, set the boolean flag `fitLastBetweenDocument-s` to true to remember the current zoom level when switching to a new document. The default value is false.
`rotateTextAnnotations`	`true`	Determines if the text inside of text annotations rotate along with the document.
`printBurnAnnotations`	`false`	Determines if VirtualViewer HTML5 should burn the annotations into the image when printing.
`printShowTypeToggles`	`false`	Includes the options to print only text or non-text annotations. If set to true, text annotations will be printed separately when the **Text** checkbox is selected. Non-text annotations will be printed separately when the **Non-text** checkbox is selected. If set to false, the Print dialog shows the Include Annotations checkbox and does not include the options to print Text and Non-text annotations separately. The default is set to false.
`exportBurnAnnotations`	`false`	Determines if VirtualViewer HTML5 should burn the annotations into the image when exporting.
`oneLayerPerAnnotation`	`false`	Creates a new annotation

Name	Default	Definition
		layer for each annotation.
`reloadDocumentOnSave`	`false`	Reloads the document model after a save. Used for systems like FileNet which generate a `documentId` on the server.
`immediatelyEditTextAnnotations`	`true`	If set to true, newly added text annotations will immediately enter 'edit' mode with the contents highlighted. Sticky note and text box annotations will be immediately placed in edit mode once drawn on the screen so that the user can edit the text after being added to the page. If you do not want newly added text annotations to immediately enter 'edit' mode with the contents highlighted, set to false.
`helpURL`	`"help/help.html";`	Passed to `window.open` when creating the help window. `window.open (helpURL,helpWindowNam- e, helpWindowParams);` This can be (and often should be) a relative URL Path.
`helpWindowName`	`"helpWindow";`	Passed to `window.open` when creating the help window. `window.open (helpURL,helpWindowNam- e, helpWindowParams);` This can be (and often should be) a relative URL Path.
`helpWindowParams`	`"scrollbars=1,width =800, height=600";`	Passed to `window.open` when creating the help window. `window.open`

Name	Default	Definition
		`(helpURL,helpWindowNam-e, helpWindowParams);` This can be (and often should be) a relative URL Path.
`fitLastBetweenDocuments`	`false`	If set to true and the default zoom mode is `fitLast`, the viewer respects that when switching between documents.
`showThumbnailPanel`	`false`	Sets the ability to hide the thumbnail panel and disable thumbnail requests to improve performance.
`showPageThumbnails`	`true`	Enables or disables the Pages thumbnail tab.
`showDocThumbnails`	`true`	Enables or disables the Documents thumbnail tab.
`showSearch`	true	Enables or disables the search tab.
`multipleDocMode`	`vvDefines.` `multipleDocModes.` `viewedDocuments`	Sets the multiple documents mode. You can use any of the following variables: `availableDocuments` - The `getAvailableDocumentId-s()` is called in the content handler to populate the list of documents. `viewedDocuments` - Documents will be added to the set of documents as the user views them during the current VirtualViewer HTML5 session. `specifiedDocuments` - Uses an array of `documentIDs` passed in as an array to

Name	Default	Definition
		`myFlexSnap.initSpecifiedDocuments()`. This is a replacement for `initViaURL()` in index.html. This is the default.
`pageManipulations`	`true`	Sets the ability to use the Page Manipulation functionality.
`pageManipulationsNewDocumentMenu`	`true`	When set to true, enables the Copy to a New Document page manipulation menu. When set to false, disables it.
`base64EncodeAnnotations`	`true`	Enables or disables Base64 encoding of annotations.
`enableRubberStamp`	`true`	When set to true, enables the Rubber Stamp functionality. When set to false, disables it.
`searchCaseSensitive`	false	Determines whether or not text searches should be case sensitive.
`createIDMAnnotations`	`false`	If set to true, creates IDM annotations. Also requires server to be configured to save IDM annotations.
`rubberStamp`	`{ textString: "Approved", fontFace: "Times New Roman", fontSize: 30, fontBold: true, fontItalic: true, fontColor: "00FF00" }, { textString: "Denied", fontColor: "FF0000" } ];`	Configure the two Rubber Stamps **Approved** and **Denied** by default.

Name	Default	Definition

`annotationDefaults`	`// Default appearance options for annotations annotationDefaults: { lineColor: "FE0000", lineWidth: 3,`
	
	`fillColor: "FE0000", stickyFillColor: "FCEFA1", // yellowish stickyMargin: 10, // also need to adjust .vvStickyNote in webviewer.css`
	
	`highlightFillColor: "FCEFA1", highlightOpacity: 0.4,`
	
	`textString: "Text",`
	
	`fontFace: "Arial", fontSize: 14, fontBold: false, fontItalic: false, fontStrike: false, // for future use fontUnderline: false, // for future use fontColor: "000000" }`
		The default appearance for a text annotation looks like a yellow sticky note. If you prefer a different look, the `annotationDefaults` config.js configuration parameter sets the default and is customizable.

Appendix B - Servlet Tags for web.xml

This appendix lists and describes all servlet tags for web.xml. The web.xml file is located in the **VirtualVieweHTML5\WEB-INF** directory.

Warning:
Please make a backup copy of the web.xml file before you edit it

The VirtualViewer HTML5 web.xml will only contain all of the Java content server web.xml parameters described in the appendix when the `contentServerType` parameter is set to integrated as shown in the following example:

Example B.1: Setting contentServerType to Integrated

```
<init-param>
<param-name>contentServerType</param-name>
<param-value>integrated</param-value>
</init-param>
```

The appendix contains the following topics:

AJAX Servlet Parameters

This table lists and describes the AJAX Servlet web.xml parameters.

Table B.1: AJAX Servlet Parameters

Name	Default	Description
`annotationOutputFormat`	`snowbound`	Enables the ability to edit and delete existing FileNet or Snowbound annotations. When set to FileNet,

Name	Default	Description
		annotations are saved in FileNet XML format. When set to Snowbound, annotations are saved in Snowbound XML format. When set to IDM, annotations are saved in the IDM format.
`contentServerType`	`integrated`	If set to integrated, the VirtualViewer HTML5 servlet will work as its own content server.

ResponseServer Parameters

This table lists and describes the ResponseServer parameters.

Table B.2: ResponseServer

Name	Default	Description
`filePath`	`.\sample-documents`	The file path the default content handler uses for retrieval and storage. Not needed when using a custom content handler.

Required Servlet Parameters

This table lists and describes the RequiredServlet servlet parameters.

Table B.3: RequiredServlet Parameters

Name	Default	Description
`tmpDir`	`N/A`	Specifies a temporary directory for files created during the processing of page manipulation routines on the server.

Optional Servlet Parameters

This table lists and describes the RetrievalServlet servlet parameters.

Table B.4: RetrievalServlet

Name	Default	Description
For AFP bit depth, please use `modcaBitDepth`.	N/A	To set the bit depth for AFP, please use `modcaBitDepth`. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
For AFP DPI, please use `modcaDPI`.	N/A	To set the DPI for AFP, please use `modcaDPI`.
For AFP format, please use `modcaFormat`.	N/A	To set the format for AFP, please use `modcaFormat`.
`baseURL`	N/A	An optional parameter that can be set when the `contentHandlerClass` is set to `com.snowbound.snapserv.servlet.FileAndURLRetriever`. The assigned value will be prepended to documentIds to create the full URL path for documents. For example, if `baseURL` is set to "http://www.snowbound.com/" and the `documentId` is "myFile.tif", the content handler will retrieve the document from http://www.snowbound.com/myFile.tif.
`bitDepth`	1	The default bits per pixel for decompression of formats not specified with individual parameters. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
`contentHandlerClass`	N/A	Name of the content handler class to use.
`defaultByteSize`	40000	Initial size of the byte array when saving to any format not TIFF or JPEG to send to VirtualViewer HTML5.
`documentCacheSize`	400000	The size in bytes of the server document cache.

Name	Default	Description
docBitDepth	1	The bit depth to use for Word documents. Valid values are 1 or 24. Must be set to 24 to display color output. Please also see wordBitDepth. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
docDPI	300	The DPI to use for Word documents. Must be set to 200 to display color output. Please also see wordDPI.
docFormat	PNG	The format to convert Word documents to. Valid values are TIFF_G4, JPEG, TIFF_LZW, PNG. Please also see wordFormat.
docxBitDepth	1	The bit depth to use for Word 2007 documents. Valid values are 1 or 24. Must be set to 24 to display color output.Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
docxDPI	300	The DPI to use for Word 2007 documents. Must be set to 200 to display color output.
docxFormat	PNG	The format to convert Word 2007 documents to. Valid values are TIFF_G4, JPEG, TIFF_LZW, PNG.
extractJpeg	true	If true, allows JPEG images to be sent directly to the client without conversion.
extractPDFPages	true	If false, the iText library will be disabled for PDF saving, resulting in raster PDFs.
extractTiffJpeg	true	If true, allows TIFF_JPEG images to be sent directly to the client without conversion.
fontMappingPath	N/A	For AFP font mapping, specifies the directory on the server of an optional

Name	Default	Description
		snbd_map.fnt file.
iocaBitDepth	1	The bit depth to use when decompressing IOCA pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
iocaDPI	200	The DPI to use when decompressing IOCA pages.
iocaFormat	TIFF_G4_FAX	The format to convert IOCA pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
jpegByteSize	600000	Initial size of the byte array when saving to JPEG to send to VirtualViewer HTML5.
jpegQuality	50	Level of quality when a page is converted to JPEG and sent to VirtualViewer HTML5.
logLevel	Finest	Detail of logging. Valid values: Severe, Warning, Info, Config, Fine, Finer, Finest, All.
modcaBitDepth	1	The bit depth to use when decompressing MO:DCA pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
modcaDPI	200	The DPI to use when decompressing MO:DCA pages.
modcaFormat	TIFF_G4_FAX	The format to convert MO:DCA pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
officeLicensePath	/Aspose.Total.Product.Family.lic	Specifies the Office 2007 plug-in licenses for Word, Excel, and PowerPoint. **Note**: If you have multiple license files for Office 2007 for Word, Excel, and PowerPoint, you need to use the officeLicensePath parameter with a comma separated list as

Name	Default	Description
		shown as the default value.
`overlayPath`	`N/A`	For AFP and MO:DCA files, specifies the path of overlays.
`pclBitDepth`	`1`	The bit depth to use when decompressing PCL pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
`pclDPI`	`200`	The DPI to use when decompressing PCL pages.
`pclFormat`	`TIFF_G4_FAX`	The format to convert PCL pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
`pdfBitDepth`	`24`	The bit depth to use when decompressing PDF pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
`pdfDPI`	`200`	The Dots Per Inch to use when decompressing PDF pages.
`pdfFormat`	`JPEG`	The format to convert PDF pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
`pixelLimit`	`N/A`	Configures server-side scaling of large raster images in order to reduce the memory footprint of the image sent to the client. If the product of an image's dimensions are greater than this number (or the product of the numbers), it is scaled to just below that. This can be expressed as a single value (i.e "1000000") or as 2 dimensions ("1000x1000").
`pptBitDepth`	`24`	The bit depth to use when decompressing PPT pages. Valid values are 1 or 24. Please note that increasing the bit depth may

Name	Default	Description
		negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
pptDPI	200	The DPI to use when decompressing PPT pages.
pptFormat	JPEG	The format to convert PPT pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.
preferencesPath	N/A	Specifies the location of stored client preferences on the server when using the default content handler.
splitAnnotationLayersByPage	false	When set to true, annotations will be saved in files by page rather than all in one file. The file name format in the default `FileContentHandler` will be as follows: `<documentName>.<layerNam-e>"-page"<pageIndex>.ann` For example: 6-Pages.tif.Default-page0.ann 6-Pages.tif.Default-page1.ann 6-Pages.tif.Default-page2.ann
supportRedactions	false	Turn on redaction support.
thumbByteEstimate	6000	The initial byte size of the buffer used on the server to transport thumbnails.
thumbnailDPI	60	Specifies the DPI to use when rendering thumbnails for vector formats such as PDF and MS Word.
tiffByteSize	40000	Initial size of the byte array when saving to TIFF to send to VirtualViewer HTML5.
vectorPDF	false	If true, PDF pages are sent to the client as vector images instead of being converted to a rasterized image. The table below shows the result for the settings for the server and/or client `vectorPDF` parameter:

Name	Default	Description		
		Server	**Client**	**Result**
		true	true	Client draws the PDF.
		true	false	Client renders the PDF.
		false	either	Server renders the PDF.
`wordBitDepth`	`24`	The bit depth to use when decompressing Word pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.		
`wordDPI`	`200`	The DPI to use when decompressing Word pages.		
`wordFormat`	`JPEG`	The format to convert Word pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG. The bit depth to use when decompressing XLS pages. Valid values are 1 or 24.		
`xlsBitDepth`	`24`	The bit depth to use when decompressing XLS pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.		
`xlsDPI`	`200`	The dots per inch to use when decompressing XLS pages.		
`xlsFormat`	`JPEG`	The format to convert XLS pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG.		

UploadServlet Parameters

Table B.5: UploadServlet Parameters

Name	Default	Description
`clearCacheOnSave`	`true`	Clears the server document cache when a document is saved.
`emailFromAddress`	`N/A`	Sets the default from address for emails sent via Email Document. This can be overridden by the equivalent parameter.
`emailServer`	`N/A`	Sets the default SMTP Server for emails sent via Email Document. This can be overridden by the equivalent parameter.

`outputConfigPath`	`N/A`	Specifies the path and name of the `output.properties` file which is used to determine the formats used when saving documents.
		
		Note: The format that you want to use may not be recognized by the default output properties. To override the default output properties and set the output format, edit the `outputConfigPath` parameter in your web.xml file as shown in the following example:
		
		`<init-param>`
		`<param-`
		`name>outputConfigPath</param-`
		`name>`
		`<param-`
		`value>/whatever/path/to/output.-`
		`properties</param-value>`
		`</init-param>`
		
		Then `output.properties` should have the following format:
		
		TIFF_LZW.format=TIFF_LZW

Name	Default	Description

		PDF.format=PDF
		JPEG.format=JPEG
		default.format=TIFF_LZW

`permanentAnnotationLinks`	`true`	WEBTOP VERSION ONLY. If false, keeps annotations from carrying over to new versions of documents.
`saveAnnotationsAsXml`	`true`	If true, saves annotations as XML rather than binary.
`sessionClass`	`N/A`	Specifies the session class to use.

Deprecated Servlet Parameters

Table B.6: ResponseServer Parameters

Name	Default	Description
`convertPDF`	`JPEG`	Specifies the format PDF pages should be converted to. (replaced by pdfFormat)
`documentCacheCount`	`1`	Number of documents the server will cache in memory. (replaced by documentCacheSize)
`jpegCompression`	`-1`	Level of quality when a page is converted to JPEG. (replaced by jpegQuality)
`maxByteMultiplier`	`20`	Maximum number of times the byte array is doubled, if the original estimate is too small, when saving to send to VirtualViewer HTML5.
`pngForPDF`	`false`	Specifies that PDF pages should be converted to PNG. (replaced by convertPDF, and then by pdfFormat)

Obsolete Servlet Parameters

Table B.7: ResponseServer Parameters

Name	Default	Description
`concurrentBWThumbs`	500	Limits the number of concurrent 1-bit thumbnail requests processed on the server.
`concurrentColorThumbs`	100	Limits the number of concurrent color thumbnail requests processed on the server.
`concurrentImages`	500	Limits the number of concurrent image page requests processed on the server.

Appendix C - JavaScript API

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

JavaScript API

Table C.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.

Appendix D - Supported File Formats

This appendix describes the file type number and read/write capabilities of all supported file formats.

VirtualViewer HTML5 is a powerful conversion tool that can transform your documents and images into many different formats. Some format types are limited in the amount of color (bit-depth) they support in an image. Some file formats read and write only black and white (1-bit deep) and other file formats support only color images (8+ bits deep). For many of these cases, VirtualViewer HTML5 automatically converts the pixel depth to the appropriate value, based on the output format specified. The chart below will help you determine whether your black and white or color document will be able to convert straight to the desired output format with no additional processing.

Table D.1: File Format Key

File Format	Description
1-bit	Black and white or monochrome images
4-bit, 8-bit, 16-bit	Grayscale images, that may appear to be black and white, but contain much more information, and are much larger than 1-bit
8-bit, 16-bit,24-bit, 32-bit	Full color images

When saving to a format, if the error returned is PIXEL_DEPTH_UNSUPPORTED (-21), the output format does not support the current bits per pixel of the image you are trying to save. The chart below will help you identify formats with compatible bit depths.

Please note that the higher the bit depth (bits per pixel), then the larger the size of the image on the disk or in memory. The higher bit depth may offer more quality, but the performance may suffer because there is a lot more image data to process. Many users may have images that appear to be black and white, however, they are stored in 24-bit color. Converting these documents to a 1-bit file format will decrease the size of the file and improve performance with no perceivable loss in quality.

If you have any questions about what format to select you may contact Snowbound Technical support on the web at http://support.snowbound.com. We do our best to support product and document specifications and to work in common platform environments, however there are always exceptions. If you find an exception please contact Snowbound Support to let us know about it.

Descriptions of Supported File Formats

Table D.2: Supported File Format Descriptions

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
AFP (MO:DCA) *	74	1, 24	1	See MO:DCA.

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
ASCII	38	1	No	Snowbound reads in ASCII text files and converts them to a bitmap.
BMP_ COMPRESSED	12	4, 8	No	Originated by Microsoft, BMP supports 1, 4, 8, and 24-bit images.
BMP_ UNCOMPRESSED	1	1, 4, 8, 16, 24	1, 4, 8, 16, 24	Originated by Microsoft, BMP supports 1, 4, 8, and 24-bit images.
BROOK_TROUT	29	1	1	Brooktrout FAX format.
CALS	18	1	1	Government specified format.
CCITT_G3	33	1	No	Group 3 compression for bitonal (1-bit) image data.
CCITT_G3_FO	53	1	No	Group 3 compression for bitonal (1-bit) image data.
CCITT_G4	34	1	No	Group 4 compression for bitonal (1-bit) image data.
CCITT_G4_FO	52	1	No	Group 4 compression for bitonal (1-bit) image data.
CFF	83	1 , 8, 24	1 , 8, 24	Compact Font Format is a lossless compaction of the Type 1 format using Type 2 charstrings. It is designed to use less storage space than Type 1 fonts by using operators with multiple arguments, various pre-defined default values, more efficient allotment of encoding values and shared subroutines within a FontSet (family of fonts).
CIMS (ABIC)	80	1	1	Check Image Management System. Developed by Carreker. Same as ABIC.
CLIP	27	1, 4, 8, 24	1, 4, 8, 24, 32	Microsoft Windows clipboard format.
COD	72	1	No	Liberty IMS black and white format.
CUT	31	8	No	Cut images are only 8 bits per pixel and the palette is stored in a separate file. Originated by Media Cybernetics.
DCS	62	32	32	The DCS format is a standard Quark Express Format. Each plane is stored as an EPS record.
DCX	11	1, 4, 8, 24	1, 4, 8, 24	Intel created this format as a multi-page .PCX format. Each page is a .PCX file in whole which can be 1, 4, 8, and 24-bit.
DIB	48	1, 4, 8, 24	No	Standard Windows Device

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				Independent Bitmap. Supports 1, 4, 8 and 24-bits. This is a multi-page file format.
DICOM	55	8, 16, 24	No	Medical image format supporting 1, 12, 16, and 24 pixel images.
DOC *	86	1, 8, 24, 32	No	Microsoft Word format. Supports Microsoft Word 97, version 8 or later. Supports 1-bit images. Cannot decompress (view) document while open in MS Word. The following features have not yet been implemented: right-to-left text flow, underlined URLs, section and paragraph borders and shading, text boxes, multi-column paragraph, Windows Meta Files (WMF) clip art, autoshapes, and embedded OLE objects. Inconsistencies exist between MS Word and the Word plugin with regards to character and line spacing. Reading support only. This is a multi-page file format.
DOCX *	93	1, 8, 24, 32	No	The .docx format is part of a family of open office XML-based formats developed by Microsoft. It is the default document format for saving applications in Microsoft Word starting with Office 2007. It is based on XML rather than Microsoft's .doc format. Reading support only. This is a multi-page file format.
DWG	90	No	24	Autodesk® AutoCAD® format. Used for computer aided design (CAD) data and metadata. The DWG format can be read in the VirtualViewer .NET Content Server.
DXF	91	No	24	Autodesk® AutoCAD® format. Used for computer aided design (CAD) data and metadata. See the following, for the full specification: http://usa.autodesk.com/adsk/serv-let

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				/item?siteID=123112&id=8446698 The DWG format can be read in the VirtualViewer .NET Content Server.
EMAIL	89	1	1	E-mail message created with MS Outlook.
EPS (preview)	14	1, 4, 8, 24	1, 8, 24, 32	Encapsulated Postscript originated by Adobe. Postscript is an interpreted language. Snowbound does not support full Postscript but will extract an embedded .TIF file in the image. Sometimes called a bitmap representation file.
EPS_BITMAP	63	8, 24, 32	1, 8, 24, 32	EPS Compressed bitmap format. It is an Adobe encapsulated Postscript file with either G4 or JPEG data embedded.
EPS_BITMAP_G4	64	8, 24, 32	1, 8, 24, 32	EPS Compressed bitmap format. It is an Adobe encapsulated Postscript file with either G4 or JPEG data embedded.
EPS_BITMAP_LZW	69	8, 24, 32	1, 8, 24, 32	EPS Compressed bitmap format. It is an Adobe encapsulated Postscript file with either G4 or JPEG data embedded.
EXCEL *	84	1, 8, 24, 32	No	Microsoft Excel Spreadsheet format for structuring and analyzing data. This is the binary file format used by Microsoft Excel 97, Microsoft Excel 2000, Microsoft Excel 2002, and Microsoft Office Excel 2003. Reading support only. This is a multi-page file format.
FileNet	78	1	1	Image format developed by FileNET Corporation for viewing documents.
FLASHPIX	54	8, 24	No	24-bit tiled JPEG format that includes multiple resolution images.
GIF	4	2, 3, 4, 5, 6, 7, 8	4, 8	Created by CompuServe for compressing 2, 3, 4, 5, 6, 7, and 8-bit palette images. Uses the LZW algorithm.
GIF_INTERLACED	44	1, 2, 3, 4, 5, 6, 7, 8	4, 8	Same as GIF except stores the raster data in an interlaced order.

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
GX2	22	4, 8	No	Originated by Brightbill Roberts for ShowPartner DOS applications. Supports 4 and 8-bit images. Simple run length encoding technique.
HTML *	82	No	24	Hyperlink Text Markup Language (HTML) is a tag-based language used to create documents for the Web. HTML forms are often used to capture information from web sites. Full HTML, Javascript and CSS support.
ICONTYPE	25	1, 4	No	Microsoft icon format. Contains a standard device independent bitmap. Supports 1 and 4 bits uncompressed.
IFF_ILBM	26	1, 4, 8, 24	1, 4, 8, 24	Used on the Commodore Amiga computers for native bitmap format. Uses a run length format for 1, 4, and 8-bit palette images.
IMG	28	1	No	Originated by Digital Research for storing 1-bit images.
IMNET	42	1	No	IMNET G4 compressed format.
IOCA (MO:DCA) *	24	1, 24	1	Image Object Content Architecture. IBM format which uses CCITT G3, G4, and IBM MMR formats. 1-bit only. This is a multi-page file format.
JBIG	71	1	1	Joint bi-level Image Experts Group. This is a highly compressed format which is stored in a TIFF header. It supports 1 or 8-bit gray scale images.
JBIG2	77	1	No	JBIG2 is a highly-compressed black and white image format that uses symbol recognition and substitution for very dramatic compression results. Snowbound's viewers and conversion programs can be used to directly view JBIG2 documents or convert those documents to a variety of output formats.
JEDMICS	56	1	No	US Military CCITT G4 tiled image format for storing Government

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				documents and drawings. Supports 1-bit per pixel.
JPEG	13	8, 24, 32	8, 24, 32	Joint Photographics Experts Group. This was a group spearheaded by Kodak for 24, 32, and 8-bit gray scale lossy compression. This is by far the best compression available for these types of images supported in the current Snowbound library.
JPEG2000 *	70	8, 24 (with plugin)	No	JPEG 2000 specification. This is similar to JPEG but produces much better compression with better quality. It is supported as a separate plugin. An option exists to set the compression level for saving.
KOFAX	23	1	No	Kofax Format.
LASER_DATA	19	1	No	Compression for documents originated by LaserData Corp. 1-bit images only.
LINE_DATA	75	1	1	Presents data for each variable on a single line.
MACPAINT	21	1	No	Original Apple bitmap file format. All MacPaint images are 720 x 576 pixels 1 bit.
MAG	61	1	No	Mag Format.
MODCA_IOCA *	49	1, 24	1	Image object content architecture. IBM format which uses CCITT G3, G4, and IBM MMR formats. 1-bit only.
MSG *	89	1	1	E-mail message created with MS Outlook.
MSP	30	1	No	Microsoft Paint program bitmap file format. Supports 1-bit images. Uses a type of RLE compression found also in compressed .BMP files.
NCR	65	1	No	A simple header with CCITT group 4 data.
ODF	98	No	No	Open Document Format is an XML-based file format for representing electronic documents such as spreadsheets, charts,

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				presentations and word processing documents.
ODP	101	No	No	Open Document Format for presentations.
ODS	97	No	No	Open Document Format for spreadsheets.
ODT	96	No	No	Open Document Format for word processing (text) documents.
OOXML	94	No	No	Office Open Extended Markup Language or Office Open XML (also informally known as OOXML or OpenXML) is a zipped, XML-based file format developed by Microsoft for representing spreadsheets, charts, presentations and word processing documents that is intended for use with the 2007 and later versions of the Microsoft Office suite.
PCL_1 (with plugin) *	57	1, 24	1	Hewlett Packard printer file format. Support for color and grayscale output. Supported as a separate plugin. This is a multi-page file format.
PCL_1 (without plugin)	57	No	1	Hewlett Packard printer file format. Support for color and grayscale output. Supported as a separate plugin. This is a multi-page file format.
PCL_5 *	76	No	1	Hewlett Packard printer file format. Support for color and grayscale output. This is a multi-page file format.
PCX	2	1, 4, 8, 24	1, 4, 8, 24	Zsoft bitmap file format. Similar to pack bits compression. Supports 1, 4, 8, and 24-bit images.
PDF(with plugin) *	59	1, 2, 4, 8, 16, 24, 32	1, 24	Portable Document Format. File format developed by Adobe to capture formatting information from a variety of desktop publishing applications. It allows the user to send formatted documents and have them appear on the recipient's monitor or printer as they were

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				intended. Compatible with the PDF/A specification and conforms to PDF v1.4. Does not currently support JPEG2000 in PDF for Java. Supports some types of Adobe specified PDF annotations, however does not support XFA annotations. Does not support corrupt PDF documents. Snowbound Software requires that the fonts needed be available on the system. This is a multi-page file format.
PDF (without plugin)	59	No	1, 24	Portable Document Format. File format developed by Adobe to capture formatting information from a variety of desktop publishing applications. It allows the user to send formatted documents and have them appear on the recipient's monitor or printer as they were intended. Compatible with the PDF/A specification and conforms to PDF v1.4. Does not currently support JPEG2000 in PDF for Java. Supports some types of Adobe specified PDF annotations, however does not support XFA annotations. Does not support corrupt PDF documents. Snowbound Software requires that the fonts needed be available on the system. This is a multi-page file format.
PDF_15	79	No	1, 24	Portable Document Format. File format developed by Adobe to capture formatting information from a variety of desktop publishing applications. It allows the user to send formatted documents and have them appear on the recipient's monitor or printer as they were intended. Compatible with the PDF/A specification and conforms

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				to PDF v1.4. Does not currently support JPEG2000 in PDF for Java. Supports some types of Adobe specified PDF annotations, however does not support XFA annotations. Does not support corrupt PDF documents. Snowbound Software requires that the fonts needed be available on the system. This is a multi-page file format.
PDF_16	92	No	1, 24	Portable Document Format. File format developed by Adobe to capture formatting information from a variety of desktop publishing applications. It allows the user to send formatted documents and have them appear on the recipient's monitor or printer as they were intended. Compatible with the PDF/A specification and conforms to PDF v1.4. Does not currently support JPEG2000 in PDF for Java. Supports some types of Adobe specified PDF annotations, however does not support XFA annotations. Does not support corrupt PDF documents. Snowbound Software requires that the fonts needed be available on the system. This is a multi-page file format.
PhotoCD	39	24	No	Kodak photo CD format. Supports only 24-bit images. This format contains at least 5 images. Get these images as you would a multi-page file format. Page 0 - 768 x 512 Page 1 - 384 x 256 Page 2 - 192 x 128 Page 3 - 1536 x 1024 Page 4 - 3072 x 2048 Images are uncompressed until the 1536 x 1024 images or greater. All images are stored as YCC data

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				which is luminance then blue and red chrominance channels. The large image must be built from the smaller images by interpolation then adding the residual data stored by Huffman encoding.
Photoshop	41	1, 4, 8, 24, 32	1, 8, 24, 32	Adobe Photoshop format for storing 1, 4, 8, 16, 24, and 32-bit images. Can be compressed or uncompressed. Images may also be stored as CMYK data or RGB.
PICT	15	1, 2, 4, 8, 16, 24, 32	1, 4, 8, 24	Apple Macintosh bitmap file format. These images may contain vector information such as lines and circles. Only the bitmap portion of data is decompressed. Uses pack bits compression. Supports 1, 2, 3, 4, 8, 16, 24, and 32-bit images.
PNG	43	1, 4, 8, 16, 24, 32	1, 4, 8, 16, 24, 32	Originated by CompuServe to replace the .GIF file format. Uses the Huffman encoding variant. Supports 1, 4, 8, 15, 16, 24, and 32-bit images. Also supports interlaced and transparency.
POWER_POINT *	85	1, 8, 24, 32	No	Microsoft PowerPoint Binary File Format which is the binary file format used by Microsoft PowerPoint 97, Microsoft PowerPoint 2000, Microsoft PowerPoint 2002, and Microsoft Office PowerPoint 2003. Reading support only. This is a multi-page file format.
PPTX *	100	1, 8, 24, 32	No	The .pptx format is part of a family of open office XML-based formats developed by Microsoft. It is the default document format for saving applications in Microsoft PowerPoint starting with Office 2007. It is based on XML rather than Microsoft's .ppt format. Reading support only. This is a multi-page file format. PPTX support requires the Snowbound

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				PDF option, the Office 2007-2010 option, and Snowbound Software's AdvancedImagingAPI.jar (provided with the product), and Java run-time environment 1.4, or 1.6 or higher. PPTX can be run in a JRE 1.5 environment if the following open source .jars are in the CLASS_PATH: retrotranslator-runtime-1.2.9.jar backport-util-concurrent-3.1.jar Aspose.slides-2.6.0.0-jdk14.jar Aspose.Total.Family.Product.Lice-nse (eval license) dom4j-1.6.1.jar log4j-1.2.16.jar AdvancedImagingAPI.jar
RAST	37	1, 8, 24	1, 8, 24	Sun raster format. Supports 1, 8, 24, and 32-bits. Run length encoded format.
RTF *	87	1, 8, 24, 32	No	The Rich Text Format is a method of encoding formatted text and graphics for easy transfer between applications. Support development in progress. This is a multi-page file format.
SCITEX	60	24, 32	24, 32	The SCITEX format is a proprietary format originated from SCITEX Corporation. Gray scale color and CMYK color images. Usually compressed.
TARGA	3	8, 16, 24, 32	8, 16, 24, 32	The SCITEX format is a proprietary format originated from SCITEX Corporation.
TARGA16	3	8, 16, 24, 32	8, 16, 24, 32	The SCITEX format is a proprietary format originated from SCITEX Corporation.
TIFF_2D	17	1	No	Tagged image file format. Created by an independent group and was supported by Aldus. .TIF files can be any number of bits per pixel, planes and several compression algorithms. The byte order may be Intel or Motorola format. The bytes

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				may also be filled from right to left or left to right. Compression may be uncompressed, pack bits, LZW, modified Huffman, CCITT G4, CCITT G3, CCITT G3-2D or JPEG. The CCITT G4 file format only saves to black and white.
TIFF_ABIC	46	4, 8	No	TIFF file with Arithmetic Binary encoding. Requires a special ABIC version of our tools. Very popular for check imaging. BW is used for 1-bit bi-level and TIFF_ABIC is for 4-bit gray scale images.
TIFF_ABIC_BW	47	1	No	TIFF file with Arithmetic Binary encoding. Requires a special ABIC version of our tools. Very popular for check imaging. BW is used for 1-bit bi-level and TIFF_ABIC is for 4-bit gray scale images. This is a multi-page file format.
TIFF_G3_FAX	8	1	1	ANSI baseline Group 3 or Group 4 compression embedded in a TIFF. This is a multi-page file format.
TIFF_G4_FAX	10	1	1	ANSI baseline Group 3 or Group 4 compression embedded in a TIFF. This is a multi-page file format.
TIFF_G4_FAX_FO	51	1	1	ANSI baseline Group 3 or Group 4 compression embedded in a TIFF. This is a multi-page file format.
TIFF_G4_FAX_ STRIP	67	No	1	ANSI baseline Group 3 or Group 4 compression embedded in a TIFF. This is a multi-page file format.
TIFF_HUFFMAN	7	1	1	TIFF file compressed using the Huffman compression algorithm. This is a multi-page file format.
TIFF_JBIG	66	1	1	Standard ANSI baseline JBIG compression embedded in a TIFF. This is a multi-page file format.
TIFF_JPEG If you have issues viewing, please see Appendix F, Troubleshooting.	40	8, 24	8, 24, 32	Standard ANSI baseline JPEG embedded in a TIFF. This is a multi-page file format.
TIFF_JPEG7	73	1, 8	1, 8	Black and white gray scale format. This is a multi-page file format.

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
TIFF LZW	9	1, 4, 8, 24, 32	1, 4, 8, 16, 24, 32	TIFF file compressed using the LZW compression algorithm. The LZW algorithm includes the look-up table of codes as part of the compressed file. This is a multi-page file format. This is a multi-page file format.
TIFF_PACK	16	1, 4, 8, 16, 24, 32	1	Simple run length encoding algorithm. This is a multi-page file format.
TIFF UNCOMPRESSED	0	1, 2, 4, 8, 16, 24, 32	1, 4, 8, 16, 24, 32	Uncompressed raw binary data. This is a multi-page file format.
WBMP	68	1	1	Windows file format for wireless devices.
WINFAX	58	1	No	A simple header with CCITT group 3 compression.
WMF	6	1, 4, 8, 24	1, 4, 8, 16, 24, 32	Microsoft Windows Metafile format. These may contain vector information such as lines and circles. Only the bitmap data is extracted. This is in the form of a standard windows DIB. May be 1, 4, 8, and 24-bit. The 4 and 8-bit images may be compressed using Microsoft RLE compression as in .BMP files.
WPG	5	1, 4, 8, 24	1, 4, 8	WordPerfect's metafile format. This is similar to the WMF file format in that it may contain vector information. Supports 1, 4, 8, and 24-bit images. Only the bitmap data is extracted.
XBM	20	1	1	Xwindows file format which encodes each pixel as an ASCII byte. Only supports 8-bits per pixel.
Xerox_EPS	45	1	No	Encapsulated Postscript for Xerox.
XLSX *	95	1, 8, 24, 32	No	The .xlsx format is part of a family of open office XML-based formats developed by Microsoft. It is the default document format for saving applications in Microsoft Excel starting with Office 2007. It is based on XML rather than Microsoft's .xls format. Reading support only. This is a multi-page

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				file format.
XPM	35	1, 4, 8	8	Xwindows bitmap file format stored as ASCII data. Each pixel is stored as an ASCII byte.
XWD	36	1, 4, 8	1, 8, 24, 32	UNIX XWD Raster format. Each pixel is stored as an ASCII byte.

* = optional only

File Type Constants Listed by File Type Number

Table D.3: Snowbound File Type Constants by File Type Number

File Type Number	File Type Name
0	TIFF_UNCOMPRESSED
1	BMP_UNCOMPRESSED
2	PCX
3	TARGA
4	GIF
5	WPG
6	WMF
7	TIFF_HUFFMAN
8	TIFF_G3_FAX
9	TIFF_LZW
10	TIFF_G4_FAX
11	DCX
12	BMP_COMPRESSED
13	JPEG
14	EPS
15	PICT
16	TIFF_PACK
17	TIFF_2D
18	CALS
19	LASER_DATA
20	XBM
21	MACPAINT
22	GX2
23	KOFAX
24	IOCA
25	ICONTYPE
26	IFF_ILBM

File Type Number	File Type Name
27	CLIP
28	IMG
29	BROOK_TROUT
30	MSP
31	CUT
32	TARGA16
33	CCITT_G3
34	CCITT_G4
35	XPM
36	XWD
37	RAST
38	ASCII
39	PHOTOCD
40	TIFF_JPEG
41	PHOTOSHOP
42	IMNET
43	PNG
44	GIF_INTERLACED
45	Xerox_EPS
46	TIFF_ABIC
47	TIFF_ABIC_BW
48	DIB
49	MO:DCA_IOCA
51	TIFF_G4_FAX_FO
52	CCITT_G4_FO
53	CCITT_G3_FO
54	FLASHPIX
55	DICOM
56	JEDMICS
57	PCL_1
58	WINFAX
59	PDF
60	SCITEX
61	MAG
62	DCS
63	EPS_BITMAP
64	EPS_BITMAP_G4
65	NCR
66	TIFF_JBIG
67	TIFF_G4_FAX_STRIP
58	WBMP
69	EPS_BITMAP_LZW
70	JPEG2000
71	JBIG

File Type Number	File Type Name
72	COD
73	TIFF_JPEG7
74	AFP
75	LINE_DATA
76	PCL_5
77	JBIG2
78	FILENET
79	PDF_15
80	CIMS
81	CIFF
82	HTML
83	CFF
84	EXCEL
85	POWER_POINT
86	DOC
87	RTF
88	PDF_LZW
89	MSG
90	DWG
91	DXF
92	PDF_16
93	DOCX
94	OOXML
95	XLSX
96	ODT
97	ODS
98	ODF
100	PPTX
101	ODP

Appendix E - Snowbound Error Codes

This appendix describes the error codes that are returned by function execution problems.

Detailed Status/Error Codes

This table lists the possible Snowbound errors and their descriptions.

Table E.1: Error Codes

Error	Error Code	Description
OUT_OF_MEMORY	-1	Failed on memory allocation. Problem with a standard memory allocation. Please see Recommended JRE Memory Settings in Appendix F, Troubleshooting for more information on the the amount of memory required.
FILE_NOT_FOUND	-2	Open call failed when trying to decompress an image.
CORRUPTED_FILE	-3	File format bad, or unreadable.
BAD_STRING	-4	String passed in is null or invalid.
BAD_RETURN	-5	Internal DLL problem. Submit a support issue at http://support.snowbound.com and attach the document you were processing when you received this error.
CANT_CREATE_FILE	-6	Fail on saving when attempting to create a new file. On this error check that you have permission to write to that directory and that there is sufficient space available on the storage device.
FORMAT_NOT_ ALLOWED	-7	Image was not recognized as a format the library can decompress.
NO_BITMAP_FOUND	-8	Getobject() call failed to return bitmap header for using DDB functions or may be returned in formats that can contain vector information such as .WPG, .WMF and .PCT if no bitmap information is found.
DISK_FULL	-9	Error writing data to the disk. Standard file i/o write failed.
BAD_DISPLAY_AREA	-10	Tried to display with negative coordinates or out of range.
PAGE_NOT_FOUND	-11	Used for multi-page file format support when attempting to access a page which does not exist. This error code provides information of an empty Word-page which is not converted to an empty page in PDF or TIFF.
DISK_READ_ERROR	-12	File format was truncated and tried to read past end of file. Standard read i/o function failed.

Error	Error Code	Description
BAD_HANDLE	-13	Application passed bad image handle. Not a valid Snowbound library image handle.
NO_CLIPBOARD_IMAGE	-14	Image not found on clipboard.
NO_SCANNER_FOUND	-15	TWAIN scanner driver not installed or not found (TWAIN.DLL).
ERROR_OPENING_ SCANNER	-16	Bad scanner driver or driver not configured properly.
CANT_FIND_TWAIN_DLL	-17	TWAIN scanner driver not installed or not found (TWAIN.DLL).
USER_CANCEL	-18	Cancel out of low level save or low level decompress. Usually not an error but termination of a function intentionally.
EVAL_TIMEOUT	-19	Date on an evaluation copy of the Snowbound product has expired.
USING_RUNTIME	-20	Version not allowed for design mode.
PIXEL_DEPTH_ UNSUPPORTED	-21	Tried to save an image to a format that does not support the image's bits per pixel. Or tried to perform an image processing function on an image whose bits per pixel is not allowed. Please see Appendix D, Supported File Formats for the pixel depths of each supported format.
PALETTE_IMAGES_ NOT_ALLOWED	-22	Some image processing operations does not work on palette images.
NO_LZW_VERSION	-23	No LZW or GIF code in this version.
JAR_NOT_FOUND_24	-24	The .jar file containing a class needed to complete the operation was not found. Please carefully examine your CLASSPATH to verify the syntax is correct, and the directory containing the RasterMaster .jar(s) is in the path. For Office 2007-2010 documents you may need to explicitly specify some .jars. If you continue to receive this error contact Snowbound Support at http://support.snowbound.com and attach the document you were processing when you received this error.
FORMAT_WILL_NOT_ OTFLY	-25	Format will not support on the fly decompression.
NO_TCOLOR_FOUND	-26	No transparency color information found.
COMPRESSION_NOT_ SUPPORTED	-27	Currently not supporting this compression format.
NO_MORE_PAGES	-28	Returned when scanning has completed all pages in the document feeder.
FEEDER_NOT_READY	-29	No more pages ready in document feeder.
NO_DELAY_TIME_ FOUND	-30	No delay time was found for the animated GIF.
TIFF_TAG_NOT_FOUND	-31	Could not find the .TIF tag.

Error	Error Code	Description
NOT_A_TILED_IMAGE	-32	Not recognized as a TIFF tiled image.
NOT_SUPPORTED_IN_ THIS_VERSION	-33	You are using a version that does not support this function. You do not have support for this file format. Please open a support issue at http://support.snowbound.com or contact your account representative to get information on the VirtualViewer option that will allow
AUTOFEED_FAILED	-34	Autofeed fail in the TWAIN Scanner.
NO_FAST_TWAIN_ SUPPORTED	-35	TWAIN driver cannot do fast transfer.
NO_PDF_VERSION	-36	The PDF processing option was not found. If you have the PDF processing option, please make sure the name of the directory containing Snowbound's pdfplug.dll is in the System environment variable Path.
NO_ABIC_VERSION	-37	No ABIC plug-in code in this version.
EXCEPTION_ERROR	-38	Internal error. An exception occurred during processing. Please enter a support ticket at http://support.snowbound.com providing the document that was being processed. If the RasterMaster function being called was not a decompress bitmap, then please include a small sample program that can be used to reproduce the issue. Please see Getting a ClassNotFoundException Error in Appendix F, Troubleshooting for more information on troubleshooting an exception error.
NO_VECTOR_ CAPABILITY	-39	No vector plug-in found in this version.
NO_PCL_VERSION	-40	The PCL processing option was not found. If you have the PCL processing option, please make sure the name of the directory containing Snowbound's pclplug.dll is in the System environment variable Path.
NO_JPEG2000_VERSION	-41	NO JPEG2000 plug-in found in this version.
SEARCH_STRING_NOT_ FOUND	-42	Did not find attempted search string.
NO_WORD_VERSION	-43	The MS Word processing option was not found. If you have the MS Word processing option, please make sure the name of the directory containing Snowbound's docplug.dll is in the System environment variable Path.
PASSWORD_ PROTECTED_PDF	-44	This file was password protected.
METHOD_NOT_FOUND	-45	The Snowbound method was not found. Please check the spelling of the method name and

Error	Error Code	Description
		Snowbound library version.
ACCESS_DENIED	-46	Access denied. Please check the security permissions.
BAD_LICENSE_ SECONDARY	-47	Primary level license loaded is bad.
PASSWORD_ PROTECTED_FILE	-48	This file was password protected for Word or other formats.
PDF_PACKAGE_NOT_ SUPPORTED	-49	This PDF file is part of a package that is not supported.
JWEBENGINE_JAR_ NOT_IN_CLASSPATH	-50	The jwebengine.jar is not in the classpath.
JAVA_1_6_RUNTIME_ REQUIRED	-51	You must be running java 1.6 or better to use HTML support.
OOXML_LICENSE_NOT_ FOUND	-52	The OOXML Aspose license file was not found.
OOXML_LICENSE_ EXPIRED	-53	The OOXML Aspose license file expired or is otherwise invalid.

General Error Define Values from Status Property

Note:
Older error define values are retrieved from the `StatusDetails` Property.

Table E.2: General Error Define Values Retrieved from Status Property

Value	Error Code	Description
GENERAL_STATUS. SYSTEM_CRASH	-100	If an internal exception is thrown, this is the resulting value.
GENERAL_STATUS. DELETE_ERROR	-101	Image data of the object failed
GENERAL_STATUS. DEFAULT	-102	What the internal values are initially set to
GENERAL STATUS. SNOWBND_OK	1	Operation completed successfully
GENERAL STATUS. SNOWBND_ERROR	-1	Operation failed. See `StatusDetails` property.
GENERAL_STATUS.	-103	Internal image data unavailable when trying to

Value	Error Code	Description
IMAGE_NOT_AVAILABLE		complete an operation
GENERAL STATUS. SNOWBND_API_NOT_ AVAILABLE	-104	API is not implemented
GENERAL STATUS. NOT_VALID	-105	Parameter is not valid
GENERAL STATUS. DISPLAY_ERROR	-106	General error display

General Status/Error Codes

This table lists the possible Snowbound general status/errors codes and their descriptions.

Table E.3: General Error Define Values Retrieved from Status Property

Error	Description
DELETE_ERROR	The image in memory cannot be removed.
DISPLAY_ERROR	Any problems with displaying an image will return this error code.
IMAGE_NOT_AVAILABLE	No image data is available to do manipulations on.
NOT_VALID	This is returned if a parameter passed into an API is not valid.
SNOWBND_API_NOT_ AVAILABLE	This is returned if an API method is not implemented in the current build.
SNOWBND_ERROR	General API error code of an unsuccessful action.
SNOWBND_OK	General API status of a successful action.
SYSTEM_CRASH	This is returned when a Critical Exception is thrown.

Appendix F - Troubleshooting

This appendix describes solutions and tips to resolve the issues that users have experienced with VirtualViewer HTML5.

Submitting a Support Issue

You may encounter an issue that is not covered by the documentation. Snowbound technical support is standing by to help you succeed. In order to get a fast, helpful response please make sure Snowbound has everything needed to reproduce the issue:

1. The configuration files: **index.html**, **config.js**, **output.properties** and **.\WEB-INF\web.xml**.
2. The document that the user is trying to view. Most issues are document specific.
3. The Java console log and the server log.
4. A list of steps that the customer took from starting the Viewer until they see the error.
5. It is helpful to have screen shot of what the user is doing when they encounter the error.
6. The version of VirtualViewer and Java that are being used.

"Please wait while your image is loaded" Message Displays Indefinitely

In some cases, images do not load in the VirtualViewer HTML5 client, and the “Please wait while your image is loaded” message displays indefinitely in the browser. This generally happens when:

1. The web server is not properly configured to handle the necessary http requests made by the client
2. The VirtualViewer server configuration itself is incorrect.

To resolve this issue, you should log the http traffic between the client and the server in order to determine which http requests are failing and why. This can be done using a browser plugin such as httpWatch (http://www.httpwatch.com) or Firebug (http://getfirebug.com). You can also use a standalone application such as Fiddler (http://www.fiddler2.com) or Wireshark (http://www.wireshark.org) which can be run independently on the client machine. For Internet Explorer 9 users, the traffic can be captured using the IE Developer Toolbar (http://www.microsoft.com/download/en/details.aspx?id=18359).

Once the http traffic has been captured, you should be able to see which requests are failing. Typically, a failed request will cause a 400 or 500 error code to be generated in the logs. Some common error codes that can occur for VirtualViewer HTML5 are as follows:

404 Not Found

This error code indicates that the requested resource on the server could not be found. This error can occur if the servlet mapping is incorrectly configured on the server. First, make sure the `servletPath` parameter value in config.js contains the correct URL mapping to the AJAX servlet. If you changed the default directory name for VirtualViewer HTML5 on the server, you will need to update this value to be consistent with that change. For more information on defining the `servletPath` parameter, please see Defining the Servlet Paths.

For VirtualViewer HTML5, the web.xml configuration should also be reviewed in addition to config.js. Make sure that the values for **<servlet-class>** and **<url-pattern>** are correct for the relative **<servlet-name>**. Please note that by default, the servlet name is set to AjaxServlet.

405 Method Not Allowed

This error code indicates that the http request contains an action (e.g. POST, GET, HEAD, etc.) that is not allowed by the requested IIS server module. With respect to VirtualViewer HTML5 .NET, this typically means that the IIS handlers for AJAXServer and **aspnet_isapi.dll** have not been properly configured in IIS. First, make sure web.config contains the following handler mapping for AJAXServer:

```
<httpHandlers>

<add verb="*" path="VirtualViewerHTML5"
type="Snowbound.VirtualViewerHTML5.AjaxServerHandler,
Snowbound.VirtualViewerHTML5" />

</httpHandlers>
```

Then, make sure that a wildcard mapping for **aspnet_isapi.dll** has been created for your website configuration. This DLL is a required resource for VirtualViewer, and is usually located in Windows under `C:\Windows\Microsoft.NET\Framework\v2.0.50727\`. To add **aspnet_isapi.dll** to your IIS configuration, please review the instructions below:

For IIS5:

- Go to **<VV web application> Properties > Directory (tab) > Configuration > "Add"**.
- For the **"Executable"** setting, provide the path to **aspnet_isapi.dll**.
- Set the **"Extension"** setting to **".*"** and left click inside the **"Executable"** path field to enable the "Ok" button below (this is a bug in IIS5... see http://support.microsoft.com/kb/317948).

For IIS6:

- Go to **<VV web application> Properties > Virtual Directory (tab) > Configuration > "Insert Wildcard application map"**, and provide the path to **aspnet_isapi.dll**.

For IIS7:

- Go to **<VV web application> Handler Mappings > Actions > "Add Wildcard Script Mapping"** and provide the path to **aspnet_isapi.dll**.

500 Internal Server Error

This error may occur if the content handler mapping is not correctly set in the web configuration. For VirtualViewer HTML5, check the `contentHandlerClass` parameter value. For VirtualViewer HTML5 .NET, check the contentHandler key value. Make sure this value contains the correct path to the content handler.

Annotation Text Does Not Appear on Separate Lines

An issue may occur where annotation text does not appear on separate lines. This occurs because Linux has different line-end characters than Windows. Linux uses just a line feed while Windows uses a carriage return + line feed (CRLF).

To solve this issue, add the following line in your code so that line-end characters will be the same on all systems:

```
System.setProperty("line.separator","\r\n")
```

Unable to Enter More Text After Using the "-" Key in an Annotation

An issue may occur where you cannot enter any more text after entering the "-" key in an annotation. This was caused by the keyboard shortcut for zoom out being defined without the CTRL modifier.

This issue will be resolved in the next release by changing the the shortcuts for zooming to the following.

- For zoom in, select CTRL+.
- For zoom out, select CTRL-.

Getting an Evaluation Period Expired Error Message When Creating a War File

An issue may occur where you receive an "Evaluation Period Expired" error message when creating a war file.

To solve this issue, look for the `servletURL` parameter in your html file. If you are using that parameter and it is pointing to an evaluation version of the servlet (possible on another machine), you will get the error messages.

Fonts Do Not Display Correctly

An issue may occur where the font displays incorrectly in the following way:

1. The text in the output document is not in the right font.
2. The text in the output document does not display the same way on Windows and on Linux.

To solve this issue, follow the steps below:

1. Inspect the document to determine what fonts it requires.
2. Make sure those fonts are installed on the system. The fonts are usually installed in the `font.properties` file.
3. Make sure the fonts are registered with Java and are of a type supported by your version of Java.

There are several resources on the Internet that discuss how to do this. There are also some helpful tools such as font viewers that make this easier. Some resources we like are:

Java Font resources:
http://mindprod.com/jgloss/font.html

Windows Font knowledgebase article:
http://support.microsoft.com/kb/918791

Java Font.Properties description from Sun:
http://java.sun.com/j2se/1.4.2/docs/guide/intl/fontprop.html

Linux Font installation:
http://linuxandfriends.com/2009/07/20/how-to-install-fonts-in-linux-ubuntu-debian

Linux Font configuration man page:
http://linux.die.net/man/5/fonts-conf

Excel 2007 xlsx files return -7 Format_not_found error

To render Word 2007, Excel 2007 and PowerPoint 2007 documents, VirtualViewer HTML5 may rely on third party packages. In order to properly integrate these packages, the CLASSPATH may have to be modified. You can specify additions to the CLASSPATH using the web.xml parameter `classPathAddition` under the `FlexSnapSIRetrievalServlet` according to the following example:

```
<init-param>
 <param-name>classPathAddition</param-name>
 <param-
value>c:\aspose\Aspose.Cells.jar;c:\aspose\;C:\aspose\dom4j-
1.6.1.jar;C:\aspose\aspose.slides-2.5.0.jar;C:\aspose\log4j-
1.2.16.jar;C:\aspose\jai_codec.jar;C:\aspose\jai_core.jar;
```

```
 </param-value>
</init-param>
```

Overlay Resources Not Pulled into APF or MODCA Document

If overlay resources such as signatures are not being pulled into an AFP or MODCA document, then make sure that the resource filename does not have a filename extension. If the resource filename has a filename extension, remove it.

Documents Slowly to Load in Multiple Documents Mode

Performance may be affected and documents may take several minutes to load if the `multipleDocMode` parameter is set to `availableDocuments` and the directory specified in the `filePath` configuration parameter (The default value is ="`C:/imgs/`".) has several hundred files. To avoid this issue, set the `multipleDocMode` parameter to `specifiedDocuments`. The default setting for the `multipleDocMode` parameter is now `specifiedDocuments`.

Images Disappear in Internet Explorer 9 when Zooming or Rotating

An Internet Explorer 9 canvas bug can cause images to disappear when you click to zoom or rotate the image.

The issue occurs because VirtualViewer AJAX is drawing faster than IE 9 can handle.

To correct this issue, we added a variable in **vvDefines.js** called `ie9DrawDelay`. It inserts a delay in milliseconds into the IE 9 drawing code which can help work around this bug.

Please add this entry to the **vvDefines.js** file:

```
ie9DrawDelay: 900,
```

The **vvDefines.js** file is located in the following directory:

```
..VirtualViewerHTML5/js/ vvDefines.js
```

The default value is 100 (100 milliseconds). The user can set the `ie9DrawDelay` variable as high as necessary. However; if it is set too high, it could cause a delay for the user each time they zoom.

The `ie9DrawDelay` variable will not work if IE 9 is set to compatibility mode. It will only work in IE 9 standard mode. Please try adding `<meta http-equiv="X-UA-Compatible" content="IE=Edge">` to **index.html** to force IE 9 standard mode.

Internet Explorer Limits URLs to 2048 Characters

Customers that included an extreme amount of parameters on their VirtualViewer command line ran out of room because the viewer infrastructure was using an HTTP GET operation. The GET URL length was limited by the Internet Explorer browser. Other browsers like Chrome and Firefox allow a much longer URL.

Please see the following work around to resolve the case where the user enters a URL longer than 2048 characters into the URL bar: For long DocumentIDs/clIDs,call `init()` instead of calling `initViaURL()`, you could simply call `init()`, then call `setDocumentId()` and `setClientInstanceId()`, followed by `openInTab()`. Please see the following example to create a Javascript function:

Instead of calling `initViaURL` in index.html, call `init`:

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

becomes

```
<body onload="virtualViewer.init()">
```

Initialize the viewer as needed:

```
myFlexSnap.setClientInstanceId("whatever their clientInstanceId is,
if any"); myFlexSnap.openInTab("whatever their document id is");
```

This solution works for all supported browsers.

Default Configuration Maximizes Performance

Please note that the default configuration for VirtualViewer HTML5 is set to maximize performance. The default settings are the following:

- The bit depth settings for vector formats such as PDF and Word are set to 1. Please note that with the bit depth set at 1 color formats will display as black and white. To view these files in color, set the bit depth to 24.
- The DPI settings for vector formats such as PDF and Word are 200. To increase the quality of an image, set the DPI to a higher value such as 400.
- The default format is set to TIFF_FAX_G4. If you are trying to view another format in color, set the format parameter to the format type.

For more information on setting parameters to maximize performance, please see Improving Performance or Quality.

Configuring to Maximize Quality

Please note that the default configuration for VirtualViewer HTML5 is set to maximize performance. If you would like to maximize quality over performance, you can change the settings as follows to maximize quality:

- Change the bit depth settings for vector formats such as PDF and Word to 24 for color documents.
- To increase the quality of an image, set the DPI to a higher value such as 400.
- The default format is set to TIFF_FAX_G4. If you are trying to view another format in color, set the format parameter to the format type.

For more information on setting parameters to maximize performance, please see Improving Performance or Quality.

Recommended JRE Memory Settings

The amount of memory required to display a document may be significantly larger than the size of the document that is stored on disk. Just like a road map, the document is folded up and compressed when it is stored. In order to see the document, it must be unfolded (decompressed) and spread out so you can see the whole map. The map takes up much more room when open for viewing. The same is true of online documents. When a document is open, a black and white letter size page at 300 dpi takes roughly 1MB of memory to display and a color page takes 25MB.

The amount of memory required to view documents varies depending on the size of the documents you are processing and the number of documents you are processing at any one time. The amount of memory needed increases as:

- You go from black and white, to grayscale, to color documents (bits per pixel increases).
- You go from compressed to uncompressed document formats (lossy compression to raw image data).
- You go from low resolution to high resolution documents (dots per inch / quality increases).
- You go from small index card size images to large blueprint size images (number of pixels increases).

Generally, higher quality documents require more memory to process. Snowbound Software does not have a one-size-fits-all recommendation for memory because our customers have such a variety of documents and different tolerances for the level of output quality. However, you can try doubling the memory available to see if that resolves the issue. Keep increasing memory until you stop getting out of memory errors. If you hit a physical or financial limit on memory, then you can do the following:

- Decrease the number of documents you have open at any one time.
- Decrease the quality of the images requested by decreasing bits per pixel, the resolution, or the size.

To calculate the amount of memory required for an image, you will need to know the size of the image in pixels and the number of bits per pixel in the image (black and white=1, grayscale=8, color=24). If you do not know the height or width in pixels, but you do know the size in inches and the dpi (dots per inch) of the image, then you can calculate the size in pixels as (width_in_inches*dots_per_inch) = width_in_pixels.

To calculate the amount of memory (in bytes), multiply the height, width and number of bits per pixel. Then, divide by 8 to convert from bits to bytes. See the following example:

(height_in_pixels * width_in_pixels * bits_per_pixel)/ 8 = image_size_in_bytes

This table lists examples of memory requirements based on image sizes.

Table F.1: Memory Requirements Based on Image Size

Image Size	Required Memory
24-bit per pixel, 640 x 480 image	640 * 480 * (24 / 8) = 921600 bytes
1-bit per pixel, 8.5" x 11" image, at 300 dpi (2550 pixels by 3300 pixels)	2550 * 3300 * (1 / 8) = 1051875 bytes
24-bit per pixel, 8.5" x 11" image, at 300 dpi (2550 pixels by 3300 pixels)	2550 * 3300 * (24 / 8) = 25245000 bytes (25 megabytes)

Displaying a Document as Landscape

If the text input document is displayed as portrait and you would like to display it as landscape, set the `ascii.attribute` parameter as shown in “Customizing the Page Layout by Setting ASCII Attribute Parameters" in Chapter 1 of the *VirtualViewer Client Administrator's Guide*.

Getting a ClassNotFoundException Error

A return status of -38 EXCEPTION_ERROR indicates an exception was thrown. This usually includes a stack trace with information about what caused the exception. If the stack trace includes **java.lang.NoClassDefFoundError:** this indicates the issue is a missing .jar file. The name of the class following **java.lang.NoClassDefFoundError:** can be used to determine which .jar file cannot be found. Check your java CLASSPATH carefully to ensure the directory containing the .jar is in the path. Please feel free to contact Snowbound Support at http://support.snowbound.com for help determining which .jar is missing.

Index