VirtualViewer®

V2.2 Java AJAX 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® Java AJAX Administrator's Guide
Part Number: DOC-0168-01
Revision: 01
VirtualViewer® for Java AJAX Release Number: 2.2
VirtualViewer® Release Number: 2.2

Printing Date: May 2012

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 **10**

Overview 10

System Requirements 10

Content Server 10

Servlet Container 10

Server Java Version 10

Client Browser Versions 10

Exceptions to Supported File Formats and Platforms 10

Packaging 10

What to Expect in an Evaluation Version of VirtualViewer Java AJAX 11

What to Expect in a Production Version of VirtualViewer Java AJAX 11

Installing the Production Version of VirtualViewer Java AJAX 11

Installing 12

Configuring 13

Configuring VirtualViewer AJAX with an Integrated Content Server 13

Customizing for Your Environment 14

Customizing Where VirtualViewer Java AJAX Gets Its Documents 14

Configuring the web.xml File 14

Configuring the config.js 15

Verifying 15

Running VirtualViewer in a Browser 15

Verifying that Your Documents Work in VirtualViewer AJAX 16

Working with the Content Handler 17

Chapter 2 - Using the VirtualViewer AJAX Client 18

The Annotation Toolbar 18

Creating Annotations 18

Moving an Annotation 19

Resizing an Annotation 19

Saving Annotations 19

Deleting Annotations 19

Undo a Deleted Annotation 20

Using Rubber Stamp Annotation Functionality 20

Using the Layer Manager 21

Creating a New Layer 23

Deleting a Layer 23

Renaming a Layer 23

Redacting a Layer 23

Printing Layers 23

The Page and Document Toolbar 24

Exporting a Document 24

Exporting a Document with Annotations 24

Sending a Document 25

Printing 25

Printing with or without Annotations 25

Zooming 26

Rubber Band Zoom 26

Fit-to-Page 27
Page Controls 27
Page Manipulation 27
Inverting 27
The Thumbnail and Docs Panels 27
Hiding the Thumbnail Panel 28
Manipulating Page Order using Thumbnails 29
Page Manipulations 29
Selecting a Page 29
Loading the Page Manipulation Context Menu 30
Cutting, Copying, Deleting and Inserting Pages 31
Saving Page Manipulations 31
Copy to New Document 31
Open Multiple Documents 34
Chapter 3 - Customizing the Configuration 36
Configuring web.xml 36
Retrieval Servlet 36
Upload Servlet 36
Defining the Servlet Paths 37
Display Your Documents 37
Configuring config.js 38
Integrated Mode 38
Customizing the User Interface 39
Turning On Buttons that are Off by Default 40

Configuring the Document Thumbnail Panel Display 40
availableDocuments 41
viewedDocuments 41
specifiedDocuments 42
Hiding the Thumbnail Panel 42
Disabling Page Manipulations 43
Disabling Copy to New Document 43
Configuring Rubber Stamp Annotation Functionality 43
Displaying the Include Annotations Checkbox 44
Export Dialog Box: Displaying the Include Annotations Checkbox 44
Print Dialog Box: Displaying the Include Annotations Checkbox 44
Turning on Redaction Support 44
Improving Performance or Quality 44
Setting the Bit Depth - xxxBitDepth 45
Setting the DPI - xxxDPI 46
Setting the Format - xxxFormat 46
Setting Office 2007 - 2010 Documents to Display Color Output 47
Default Configuration Maximizes Performance 48
Configuring to Maximize Quality 49
Clearing a Document from the Cache 50
Defining the Number of Pages Cached in Memory 50
Chapter 4 - Using Advanced Features 51
Virtual Documents 51
Loading Virtual Documents 51

Virtual Document Syntax 51
Displaying a Virtual Document 52
Printing Virtual Documents 52
Annotation Security: Watermarks and Redactions 52
The Annotation Security Model 52
Permission levels 52
Level Definitions 53
Retrieving Annotation Layers 54
Key/Value Pairs 54
Saving Redaction Layers 55
Printing Layers 55
FileNet Annotations 55
Annotation Mapping 56
Connecting Your Document Store 56
What is the Content Handler? 57
How the Content Handler Works 57
Plugging in a Custom Content Handler 57
FlexSnapSIContentHandlerInterface 58
CacheValidator 58
CacheValidator Method Detail 58
Extracting Parameters from ContentHandlerInput 59
Populating Parameters for ContentHandlerInput 60
Populating Parameters for ContentHandlerResult 60
How to Return an Error for Display in the Client 61

Content Handler Methods 61
VirtualViewerContentHandlerInterface Method Detail 63
VirtualViewerSaverInterface Method Detail 71
Using KEY_ANNOTATION_LAYERS 74
Appendix A - Config.js Parameters 77
Descriptions of Config.js Parameters 77
Appendix B - AJAX Servlet web.xml Parameters 82
Description of the AJAX Servlet Parameters 82
Appendix C - Servlet Tags for web.xml 83
ResponseServer Servlet Parameters 83
Required Servlet Parameters 83
Optional Servlet Parameters 84
UploadServlet Servlet Parameters 87
Deprecated Servlet Parameters 88
Obsolete Servlet Parameters 88
Appendix D - JavaScript API 89
JavaScript API 89
Appendix E - Supported File Formats 90
Descriptions of Supported File Formats 90
File Type Constants Listed by File Type Number 103
Appendix F - Snowbound Error Codes 106
Detailed Status/Error Codes 106
General Error Define Values from Status Property 109
General Status/Error Codes 109

Appendix G - Troubleshooting **111**

"Please wait while your image is loaded" Message Displays Indefinitely 111

404 Not Found 111

405 Method Not Allowed 111

500 Internal Server Error 112

Annotation Text Does Not Appear on Separate Lines 113

Unable to Enter More Text After Using the “-” Key in an Annotation 113

Getting an Evaluation Period Expired Error Message When Creating a War File 113

Fonts Do Not Display Correctly 113

Excel 2007 xlsx files return -7 Format_not_found error 114

Overlay Resources Not Pulled into APF or MODCA Document 114

Documents Slowly to Load in Multiple Documents Mode 115

Default Configuration Maximizes Performance 115

Configuring to Maximize Quality 115

Recommended JRE Memory Settings 115

Displaying a Document as Landscape 117

Submitting a Support Issue 117

Chapter 1 - Getting Started

Overview

Snowbound Software's Java AJAX 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, **VirtualViewerJavaAJAX.zip**. This zip file installs all of VirtualViewer Java AJAX components. For information on configuring VirtualViewer Java AJAX, please see Chapter 3, "Customizing the Configuration".

System Requirements

This section describes the system requirements to run VirtualViewer Java AJAX.

Content Server

The VirtualViewer Java AJAX Server requires the VirtualViewer Java Content Server in order to function.

Servlet Container

The VirtualViewer Java AJAX Server 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

The VirtualViewer Java AJAX Server requires a JRE of at least 1.5 or higher.

Client Browser Versions

The supported browsers are Internet Explorer 6, 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 insure 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 www.support.snowbound.com to let us know about it.

Packaging

VirtualViewer Java AJAX is delivered as a .zip file including the **VirtualViewerJavaAJAX.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.

- **VirtualViewerJavaAJAXAdminGuide.pdf:** This guide describes how to use and configure VirtualViewer Java AJAX Client.
- **VirtualViewerJavaAJAXViewerReleaseNotes.pdf:** The release notes describe the latest additions and improvements to VirtualViewer AJAX.

What to Expect in an Evaluation Version of VirtualViewer Java AJAX

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 Java AJAX

When you purchase VirtualViewer Java AJAX, you will receive a set of fully licensed binary files. The files will include **VirtualViewerJavaAJAX.zip**.

Installing the Production Version of VirtualViewer Java AJAX

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.

If you are upgrading from an older version of VirtualViewer AJAX to version 2.1 or more recent, you may find that the structure of the directories have changed. This is on purpose to improve performance. Previously, we separated the two directories VirtualViewerJavaAJAXServer and VirtualViewerJavaContentServer as a demonstration to show how the content server can be separated out so it can be run independently from the VirtualViewer servlet functionality. In the

new version, both web.xml files are combined for simplicity, clarity, and better performance. This is known as the integrated mode configuration.

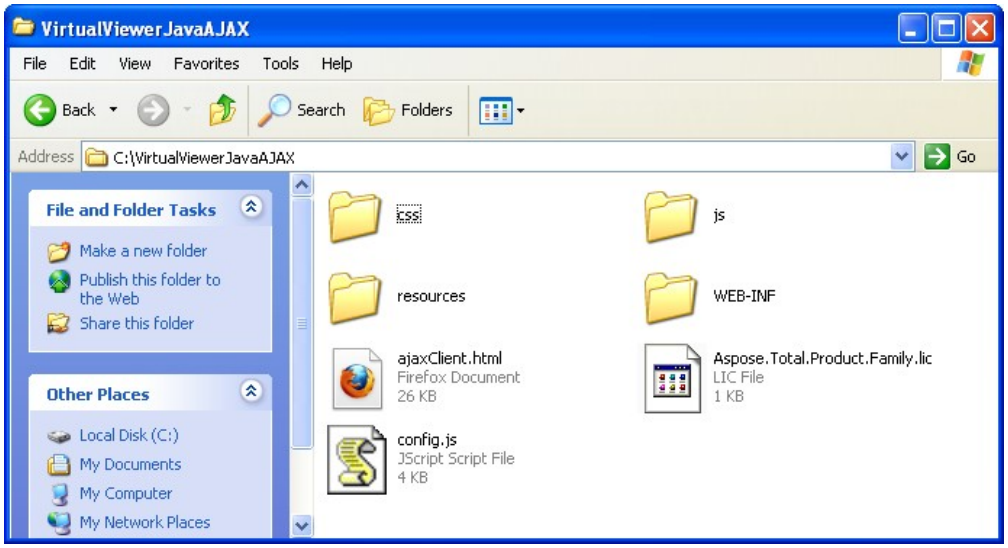
We still support separating the AJAX Server and Content Server onto different machines or into different directories (non-integrated or http mode). Please contact Snowbound Software Support at www.support.snowbound.com if you need assistance.

Installing

To install VirtualViewer Java AJAX, follow the steps below:

1. Extract the **VirtualViewerJavaAJAX.zip** file to a directory.
2. The extracted .zip file includes the **VirtualViewerJavaAJAX.war** file.
3. Save the **VirtualViewerJavaAJAX.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.
4. Extract the files in the **VirtualViewerJavaAJAX.war** file.
5. Select a location to extract the .war files to.
6. In the **VirtualViewerJavaAJAX** directory, you will see the extracted files for VirtualViewer Java AJAX.



7. Find the web application (webapps) directory where you want to install the files.

   For example:

   `C:\Tomcat 6.0\webapps.`

8. From the extracted zip directory, copy the `VirtualViewerJavaAJAX` directory to your `webapps` directory.

Configuring

This section describes the configuration that you need to do to run VirtualViewer Java AJAX. For the client, server, and content server to work together properly, you need to set the parameters described below. These parameters set up the default content handler and should be modified to point to your customized content handler.

Configuring VirtualViewer AJAX with an Integrated Content Server

VirtualViewer AJAX is configured to have the AJAX server and the content server in the same directory on the same machine. This is known as *integrated mode*. Snowbound recommends this configuration for performance and maintainability.

VirtualViewer 2.0 and previous had a multi-directory configuration known as http integration. If you are installing a previous version of VirtualViewer Java AJAX with more than one server directory such as the **`VirtualViewerJavaAJAXServer`** directory and the **`VirtualViewerJavaContentServer`** directory, follow the instructions below:

1. Find the web application (webapps) directory where you want to install the files.

   For example:

   `C:\Tomcat 6.0\webapps.`

2. From the extracted zip directory, copy the **`VirtualViewerJavaAJAXServer`** directory to your **`webapps`** directory.

3. From the extracted zip directory, copy the **`VirualViewerJavaContentServer`** directory to your **`webapps`** directory.

In the web.xml file, for the `AjaxServlet`, indicate that you want the `AjaxServlet` to serve as its own content server.

Change the value of the `contentServerType` parameter from the default value of http to integrated:

Example 1.1: Configuring Integrated Mode

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

Once the `AjaxServlet` is configured to run in integrated mode, you should then also make sure to update any relevant content server related init-params. Most importantly, this would include the `ContentHandler` parameters such as the following:

Example 1.2: Setting Updating the Content Server Related init-params

```
 <init-param>
       <param-name>contentHandlerClass</param-name>
 <param-value>com.snowbound.snapserv.servlet.FileContentHandler</param-
value>
</init-param>
 <init-param>
       <param-name>contentHandlerClass</param-name>
 <param-value>com.snowbound.snapserv.servlet.FileContentHandler</param-
value>
</init-param>
 <init-param>
       <param-name>filePath</param-name>
 <param-value>/Users/imgs/</param-value>
</init-param>
```

Customizing for Your Environment

This section is a summary of what you must do next to get to a production quality VirtualViewer Java AJAX and optional items running.

Customizing Where VirtualViewer Java AJAX Gets Its Documents

A sample **web.xml** file is supplied when the application is installed, and it is located in the **VirtualViewerJavaAJAX/WEB-INF** directory.

Configuring the web.xml File

The **web.xml** file included in the **`WEB-INF`** directory needs to be edited as shown in the examples below to specify the location of the content server and the AJAX server.

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

The `codebase` parameter specifies where the AJAX Server is running.

Example 1.3: Setting the codebase Parameter to Specify Location of AJAX Server

```
<init-param>
 <param-name>codebase</param-name>
 <param-value>http://localhost:9080/Workplace
```

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

The `servletURL` parameter specifies where the Content Server is running.

Example 1.4: Setting the servletURL Parameter to Specify the Location of the Content Server

```
<init-param>
 <param-name>servletURL</param-name>
 <param-value>http://localhost:9080/Workplace
</param-value>
</init-param>
```

Configuring the config.js

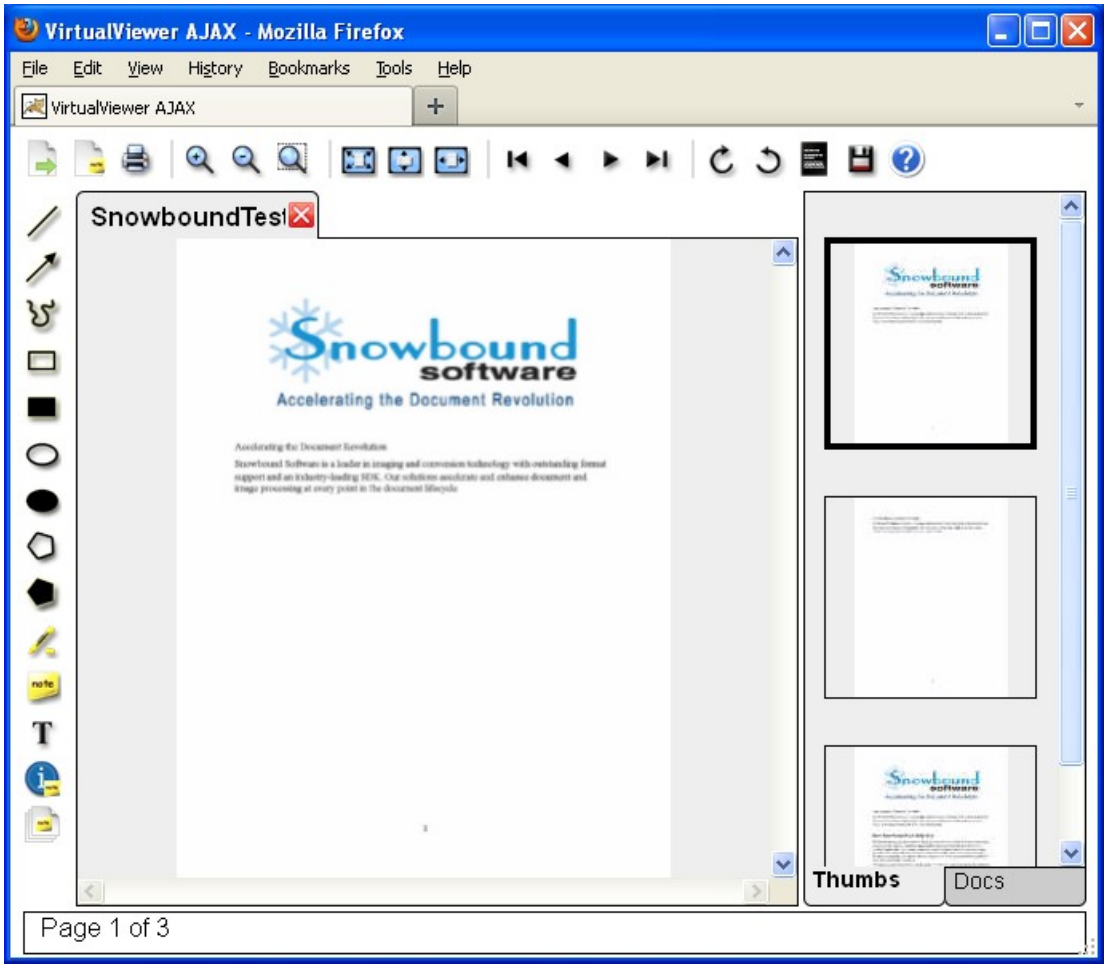
The `config.js` file located in the **`VirtualViewerJavaAJAX`** directory needs to be edited to specify the location of the AJAX Servlet.

```
var servletPath =  "/VirtualViewerJavaAJAX/AjaxServlet";
```

Verifying

Running VirtualViewer in a Browser

Once all components have been installed, VirtualViewer AJAX Client will start up from any supported browser. No client components are needed on the client machine. The following example shows VirtualViewer AJAX Client loaded in a browser:



Snowbound Software provides some sample documents in the VirtualViewer AJAX installation to get you started. Copy the files in the sample-documents subdirectory into directory specified in the `filePath` parameter .The specified directory is **`c:/imgs`** by default.

If you are able to see the three documents that came with your VirtualViewer .NET AJAX installation, then you have successfully installed it.

Verifying that Your Documents Work in VirtualViewer AJAX

To view your own images in VirtualViewer Java AJAX, you must put the document that you want to view in the `c:/imgs` directory or the directory that the `filepath` parameter specifies in the web.xml file. For information on configuring VirtualViewer Java AJAX, please see Chapter 3, “Customizing the Configuration".

```
<param name="filePath" value="C:/imgs/"/>
```

You must then append the parameter `documentId` to the end of the URL in order to specify the ID of the document you want to display. For example, if you want to display the file named commerce.tif, add that name to after `documentId` as shown in the following example:

Example 1.5: Setting Specifying the Document to Display

```
http://server:port-
/VirtualViewerJavaAJAXServer/ajaxClient.html?documentId=filename.ext
http://server:port-
/VirtualViewerJavaAJAXServer/ajaxClient.html?documentId=commerce.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.

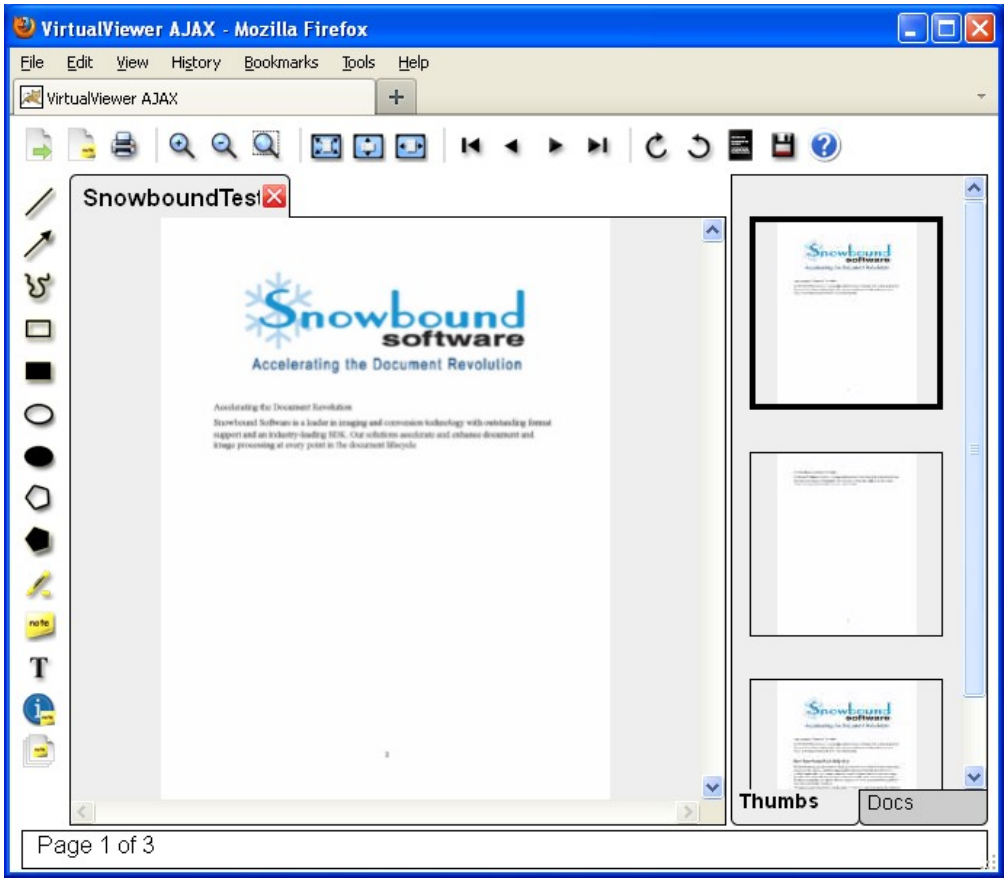
Working with the Content Handler

The VirtualViewer content handler is a Java class that the serlvet will call on to perform various actions concerning the retrieval and storage of content. By default, the VirtualViewer 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 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. For more information, please see Connecting to Your Document Store.

Please see the next topic Chapter 2 - Using the VirtualViewer AJAX Client.

Chapter 2 - Using the VirtualViewer AJAX Client

This chapter describes the available functionality and features in the VirtualViewer AJAX Client.



The Annotation Toolbar

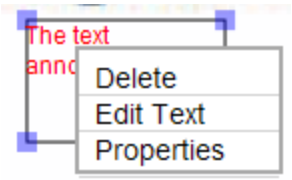
Creating Annotations

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: line, arrow, freehand, rectangle, filled rectangle, ellipse, filled ellipse, polygon, filled polygon, highlight, sticky note, and text.

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

You can also click on the annotation and then right-click on your mouse to display a contextual menu. The contextual menu contains the following menu items:

- **Delete** - Deletes the annotation.
- **Edit text** - Displays a dialog box to edit the text with an OK or Cancel button. The menu item is only available for Text or Sticky Note annotations.
- **Properties** - Opens the Annotations Properties.



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

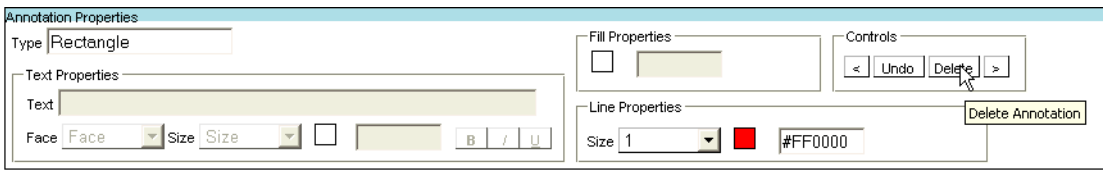
To resize an annotation, click on it until it is highlighted and selection squares display on each of the annotation's corners. Drag one of the selection squares to resize the annotation to the desired size.

Saving Annotations

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

Deleting Annotations

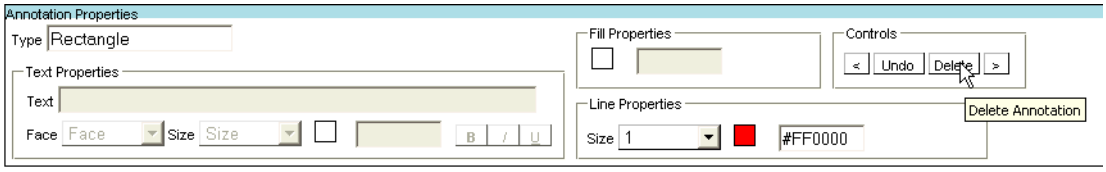
To delete an annotation, click on the annotation properties icon to open annotation properties  . Click on the annotation to select it. In the controls section of the annotation properties window, select the **Delete** button.



You can also right-click on your mouse to display a contextual menu. From the contextual menu, select **Delete**.

Undo a Deleted Annotation

To undo a deleted annotation, select the **Undo** button in the controls section of the open annotation properties window

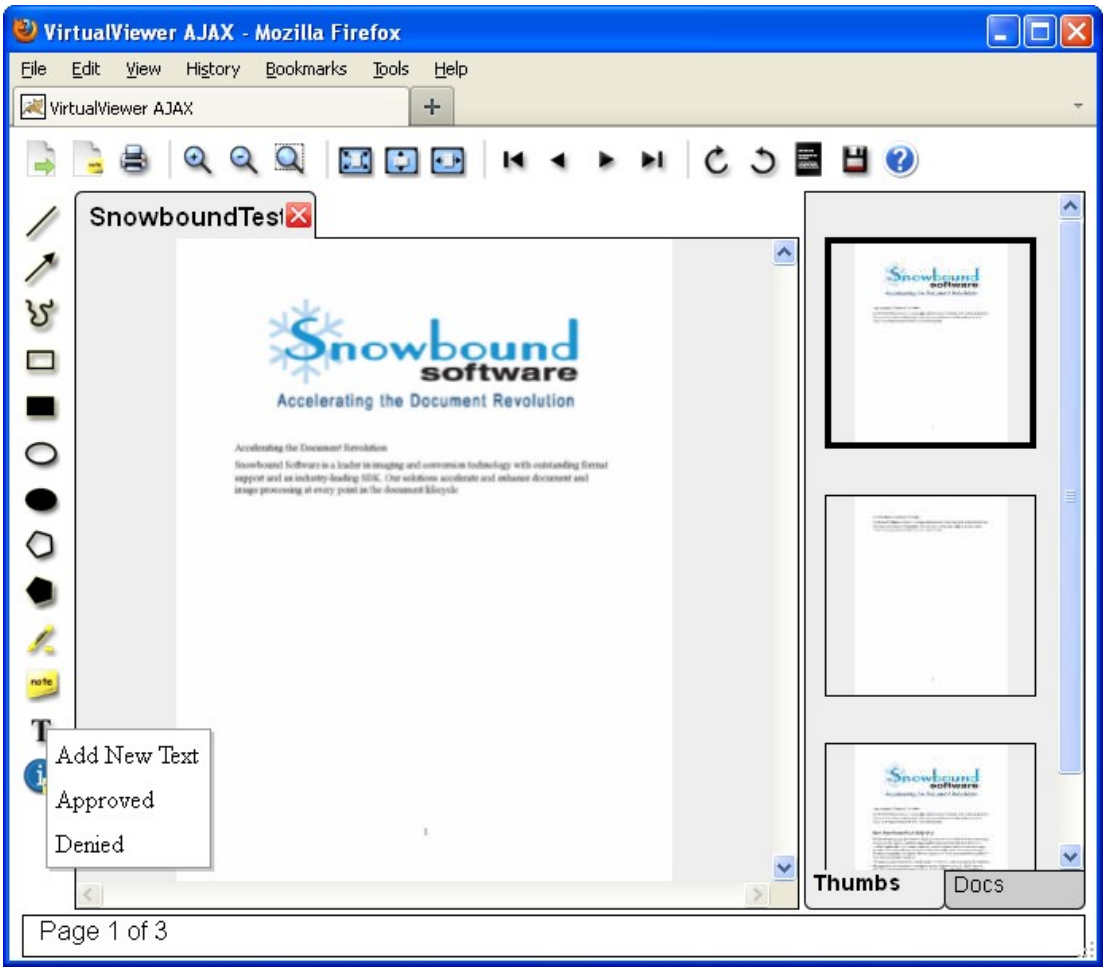


Using Rubber Stamp Annotation Functionality

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".

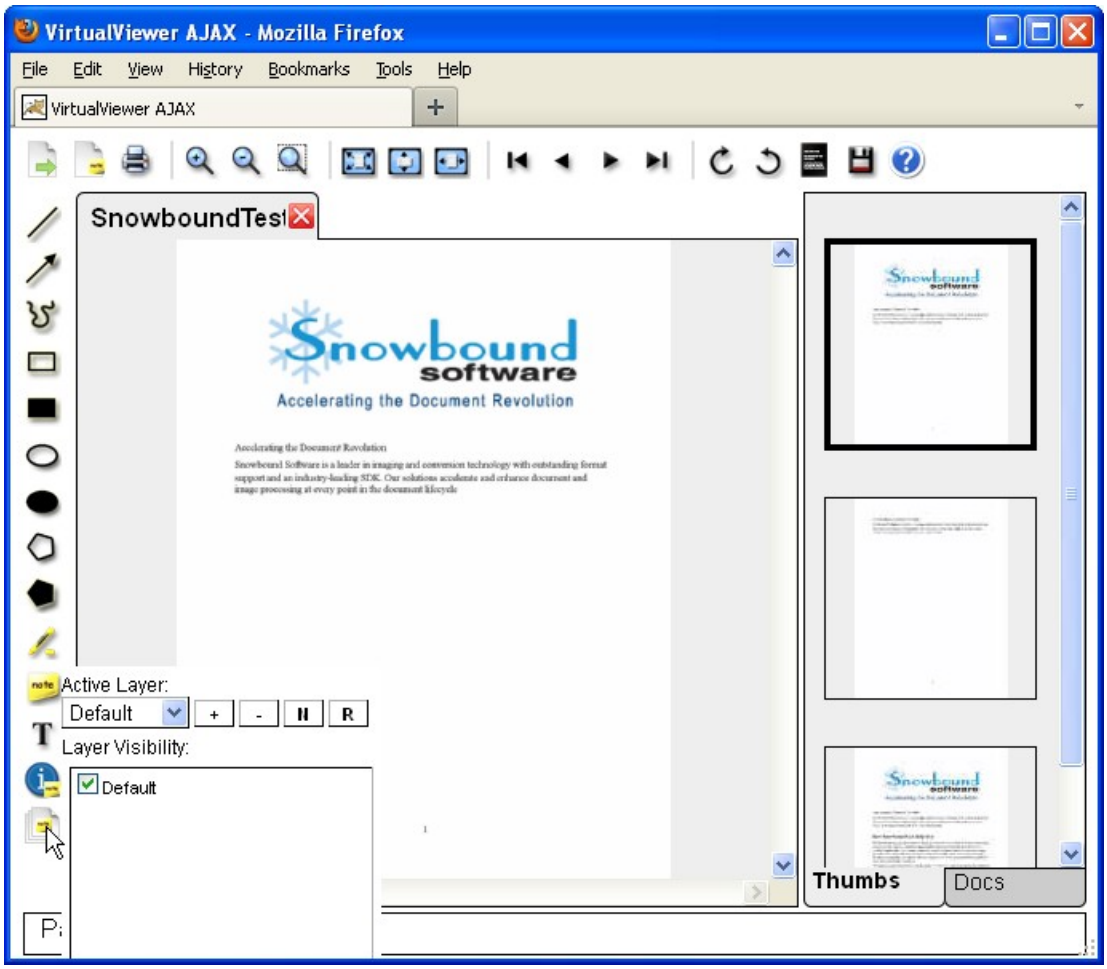
If the `enableRubberStamp` parameter is set to true and one or more Rubber Stamps are defined, then clicking on the **Text Edit** annotation toolbar button as shown below will produce a menu allowing you to select a Rubber Stamp from the available options or to Add New Text to add a traditional text annotation:

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

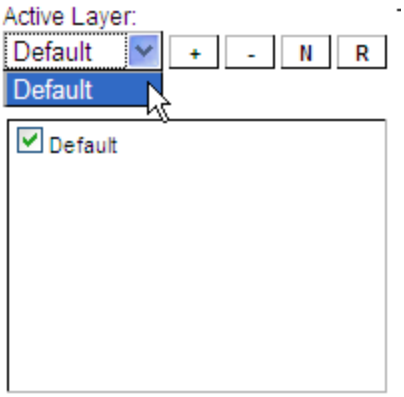
To use the layer manager, select the Layer Manager button. 



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 check marks.

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

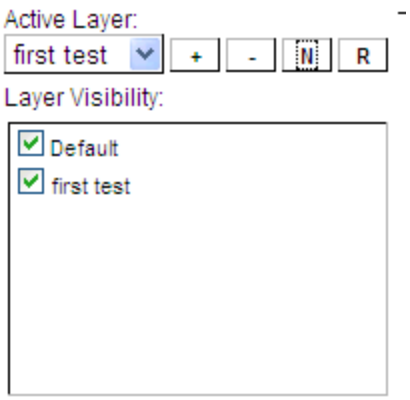
Select **OK** to delete the layer.

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

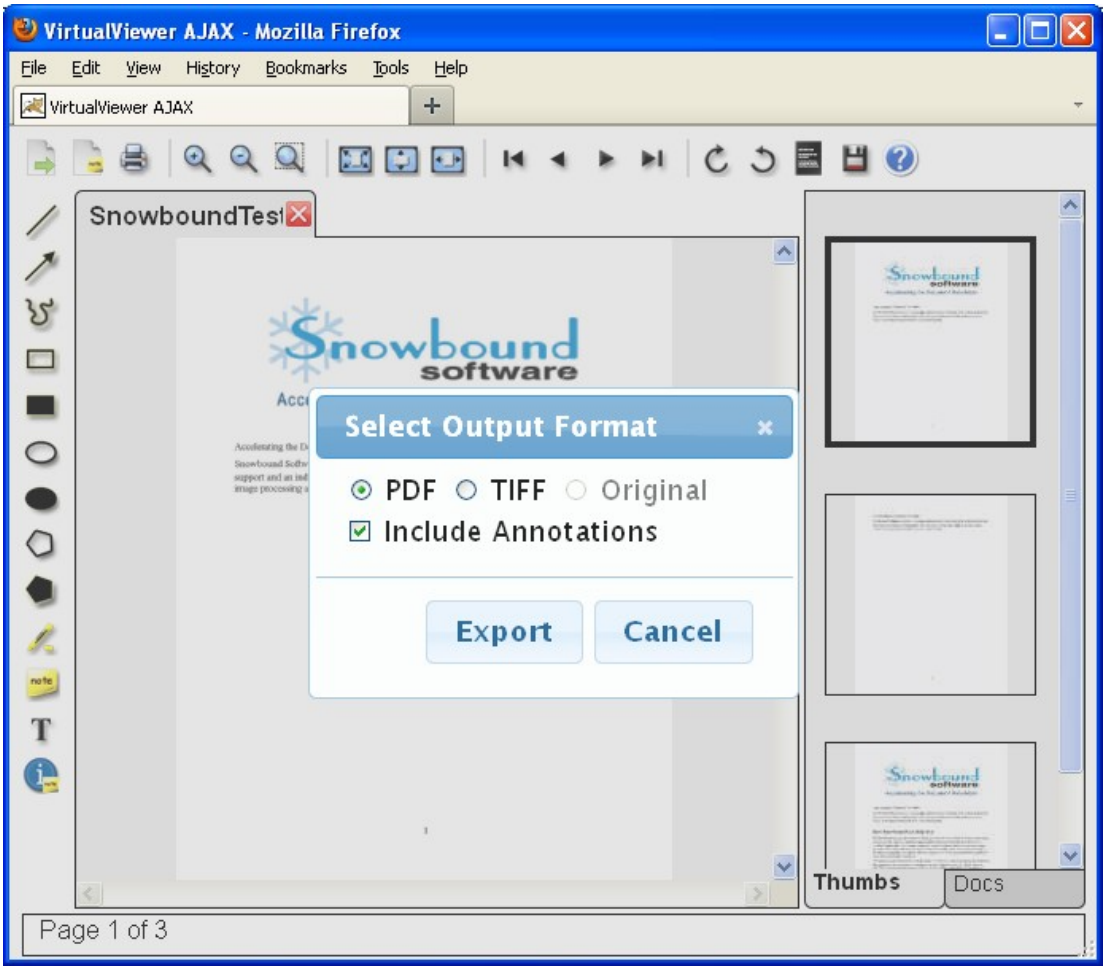
Exporting a Document

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 includes 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".



Sending a Document

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

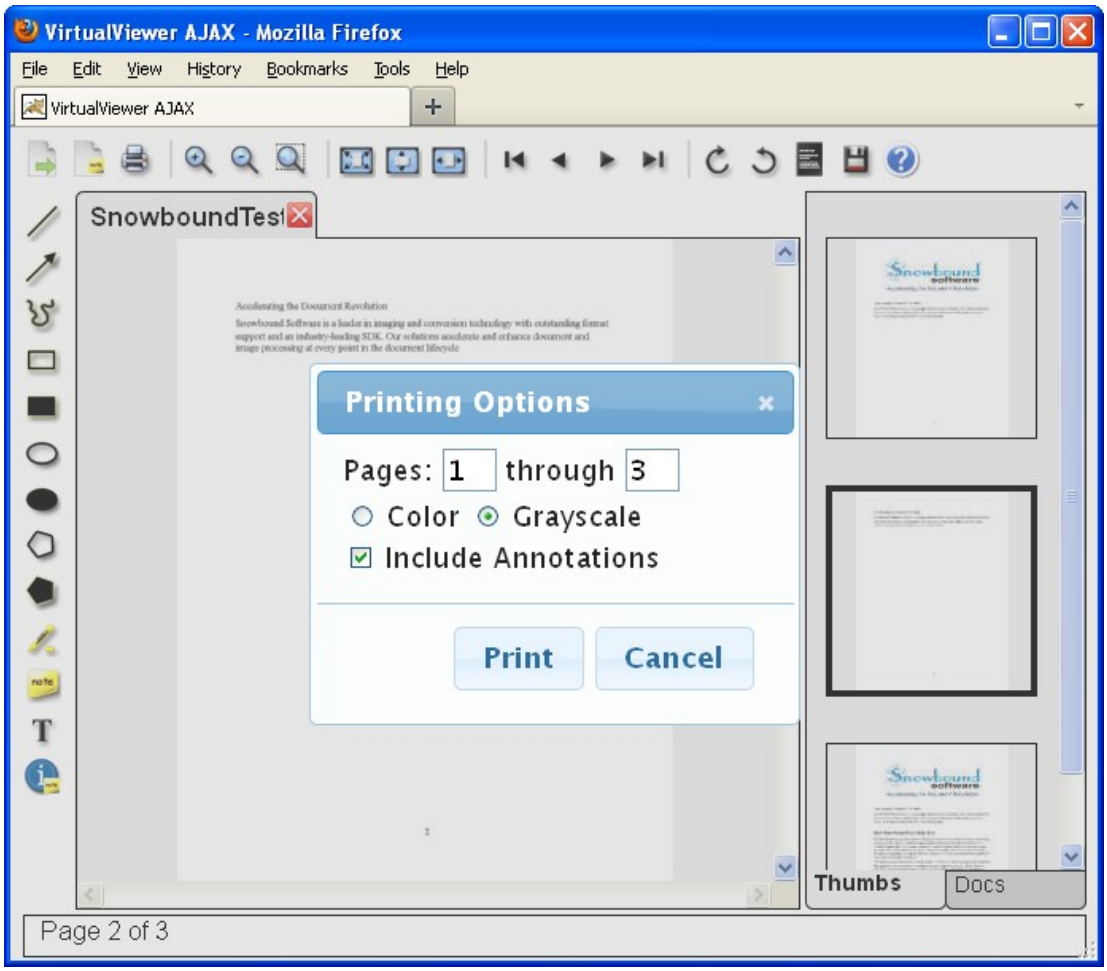
Printing

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

Printing with or without Annotations

The print dialog box includes the Include Annotations checkbox to select the option to print with or without annotations. Annotations will only be included when the Included Annotations check-box 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 Print Dialog Box: Displaying the Include Annotations Checkbox in Chapter 3, "Customizing the Configuration".



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  , Rotate Counter-clockwise, Flip Horizontal  and Flip Vertical. 

To turn the Flip Horizontal and Flip Vertical buttons back on, please see Turning On Buttons that are Off by Default.

Inverting

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

The Thumbnail and Docs 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 **Thumbs** tab to display the thumbnails for the current image being viewed. Select the **Docs** 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.

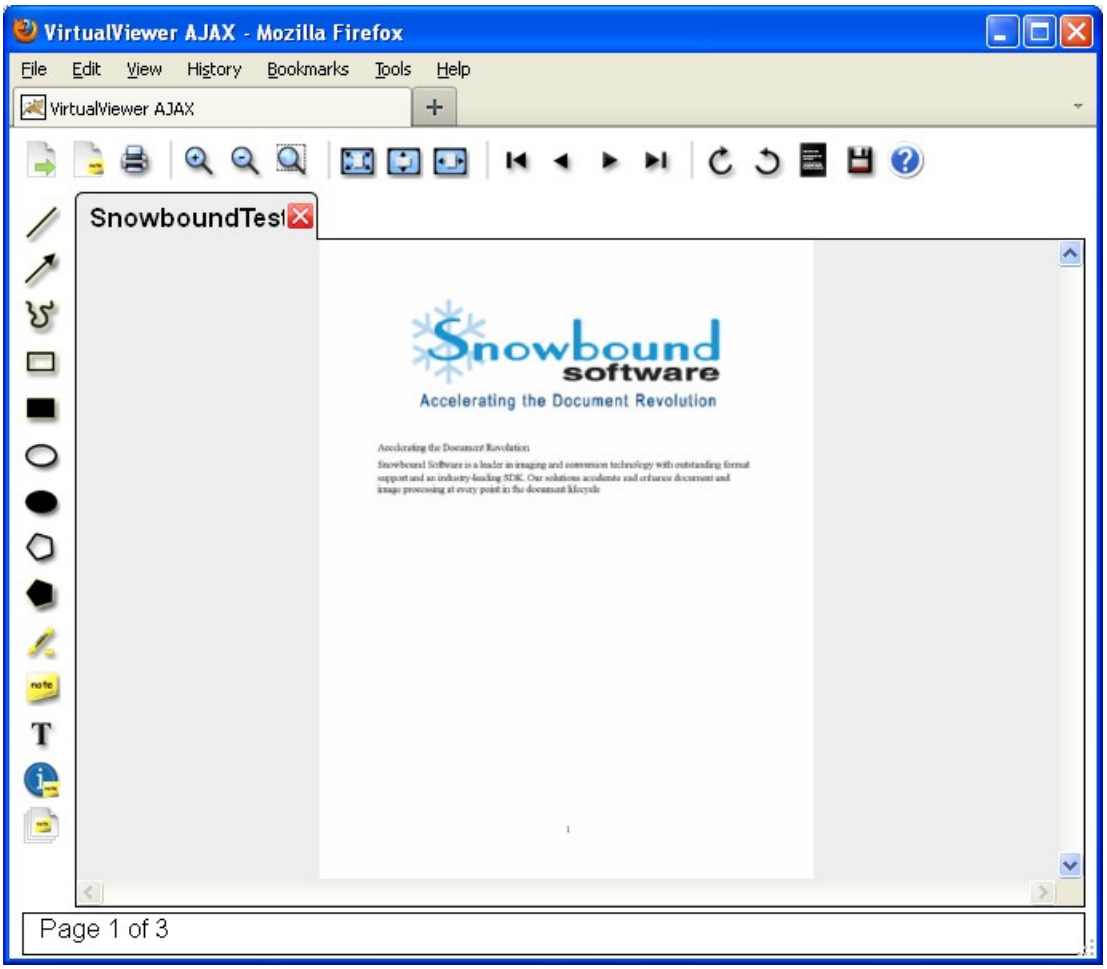
Hiding the Thumbnail Panel

The Thumbnail panel provides a convenient way to:

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

However, this convenience does have a price. VirtualViewer performance degrades because it is processing every page in the document Thumbs panel and/or the first page of every document in the Docs 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".

The following shows VirtualViewer AJAX with the Thumbnail Panel hidden:



Manipulating Page Order using Thumbnails

VirtualViewer AJAX 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.

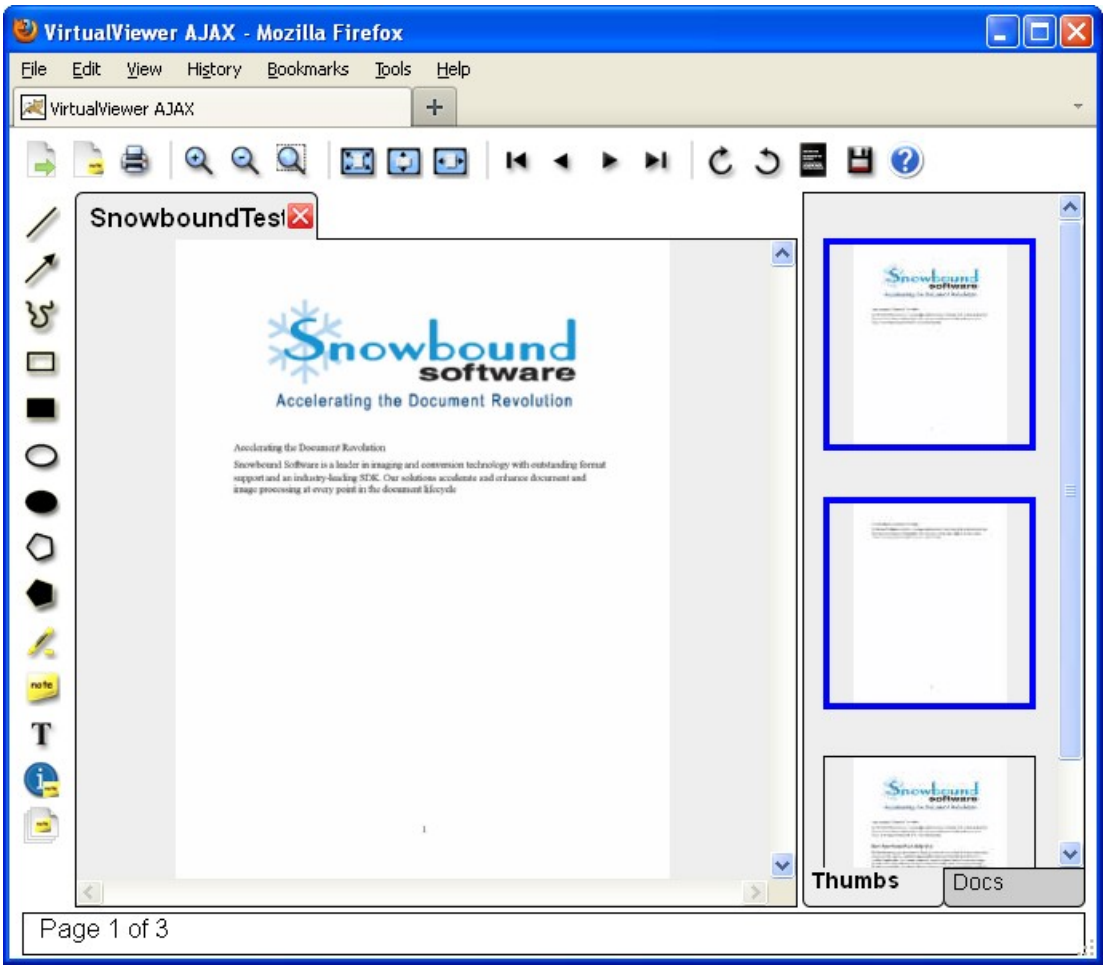
Page Manipulations

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

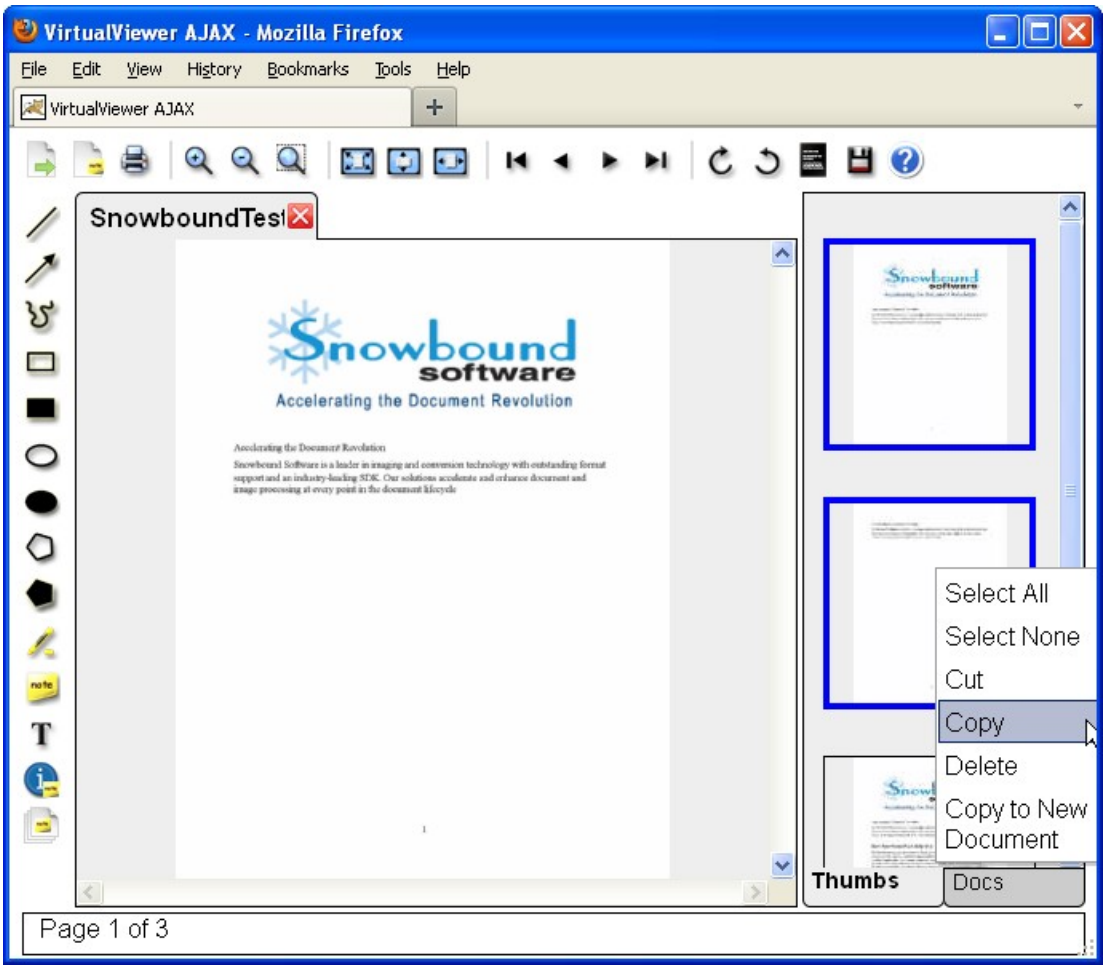
To select a page for page manipulation, left click on a page thumbnail in the Thumbs tab. A blue selection border around the thumbnail indicates that it has been selected for page manipulation.

- 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 AJAX.

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

Saving Page Manipulations

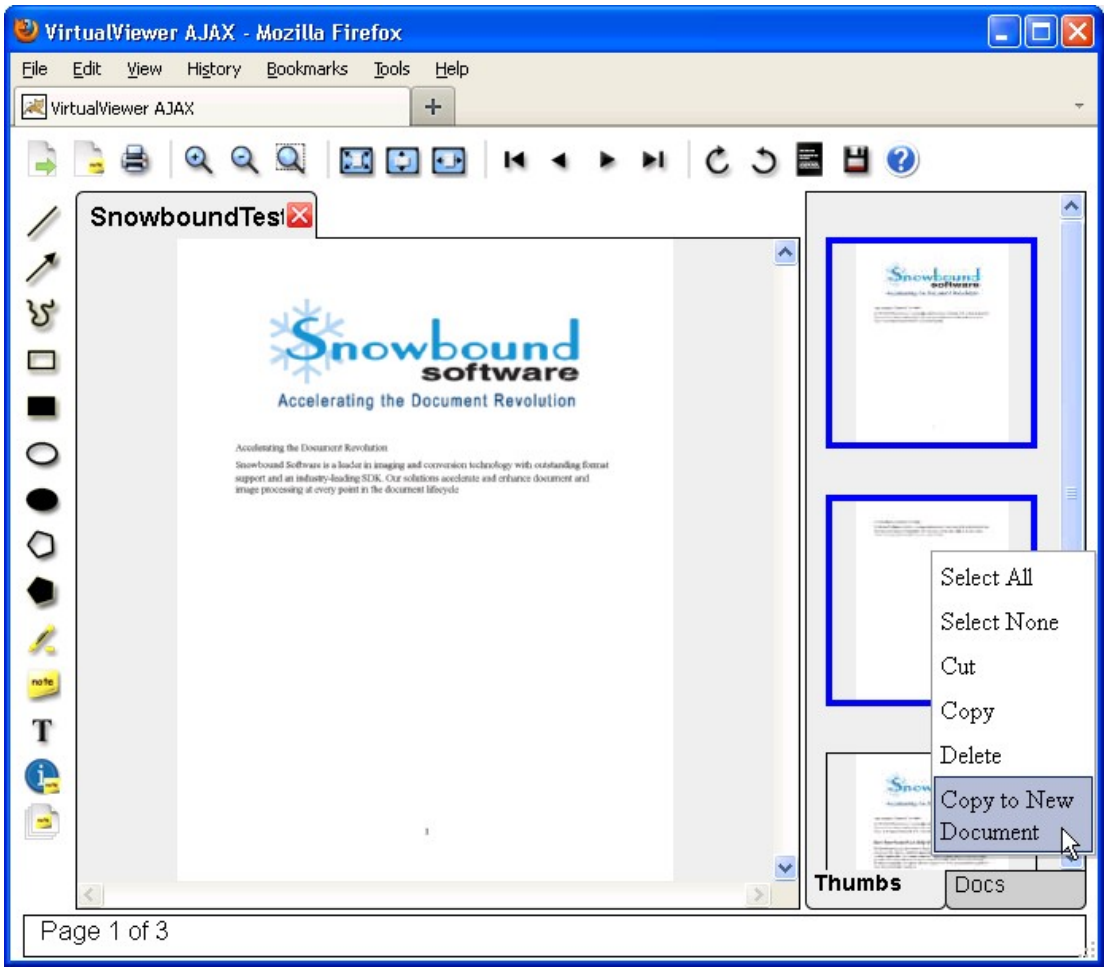
Select **Save** to save page manipulations, including rotations and inversions, to the file currently being viewed.

Copy to New Document

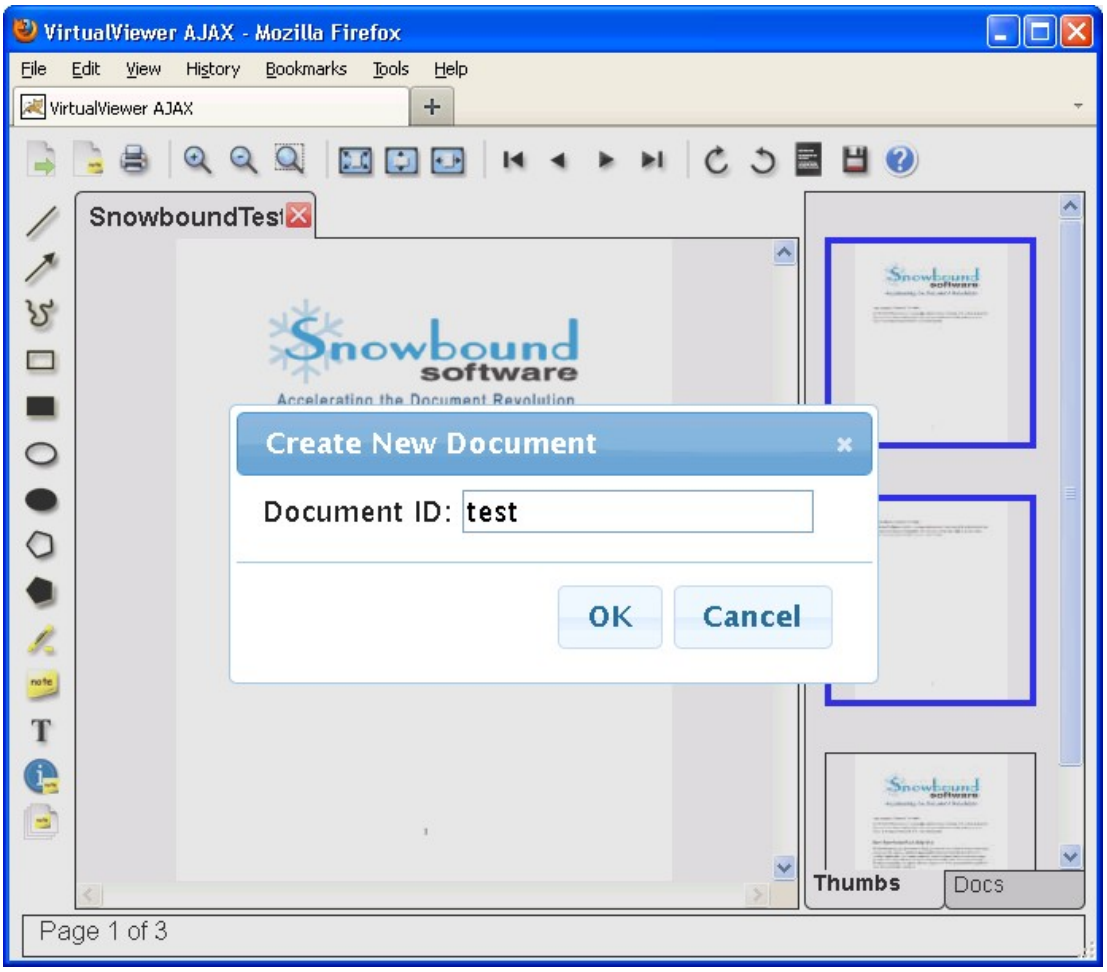
To copy to a new document, follow the steps below:

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.

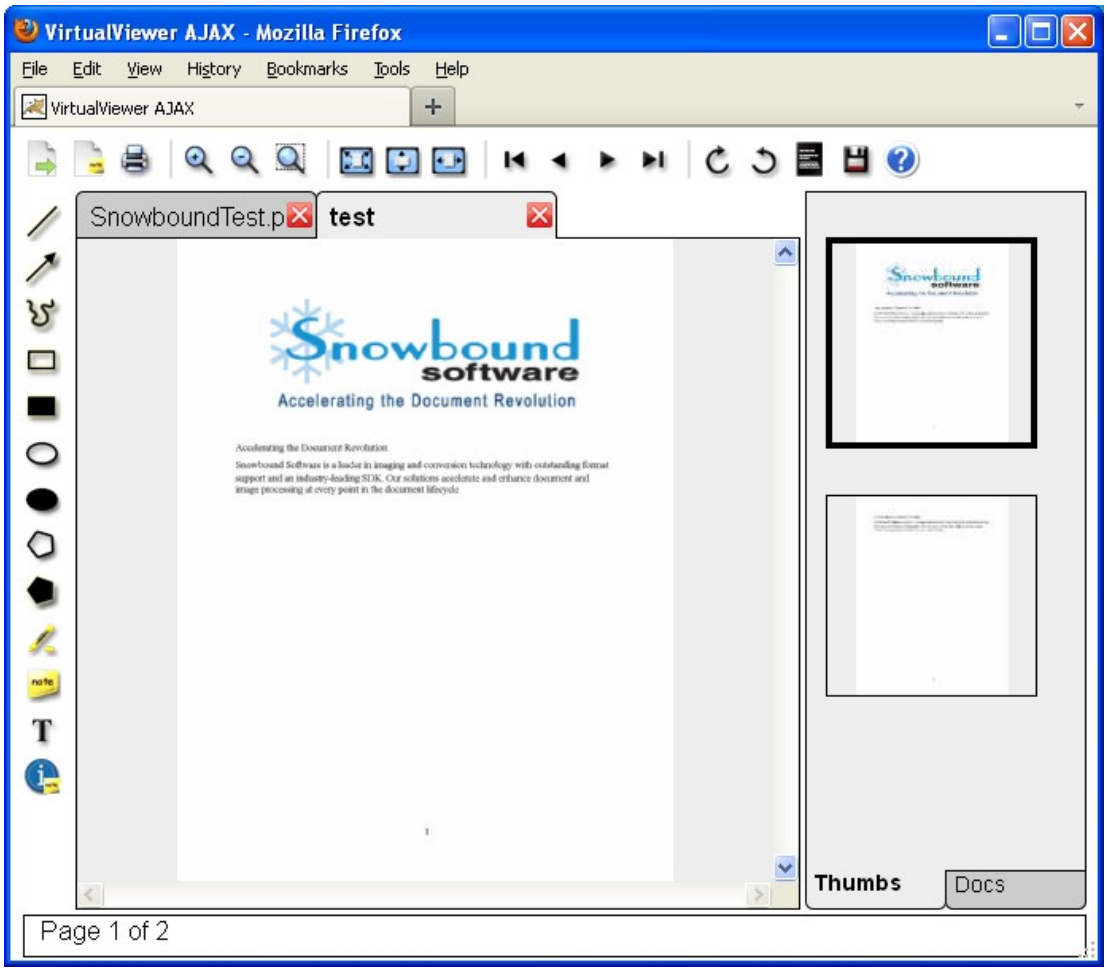


3. In the Create New Document window, enter the new document name in the Document ID field and select the **OK** button.



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

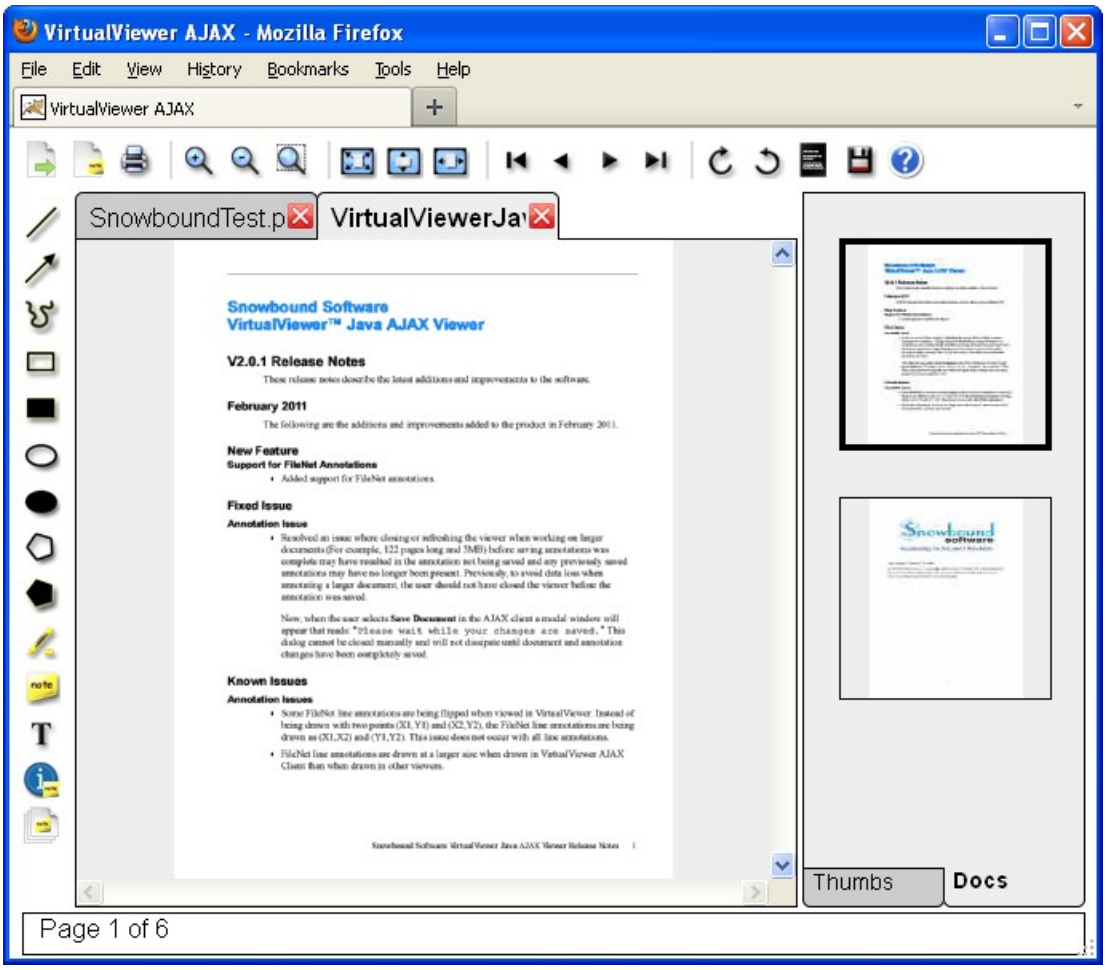
To open multiple documents, select the **Docs** tab located at the bottom of the thumbnail panel. From the Docs pane, simply left-click any document you would like to open in the main viewer. When a new document has been clicked in Docs pane, it will display in the main viewer and have a new document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

You can open multiple documents at the same time using one of the three available multiple document modes:

- available documents
- viewed documents
- specified 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 "Troubleshooting".



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

Chapter 3 - Customizing the Configuration

This chapter shows how to configure VirtualViewer Java AJAX on your system.

Configuring web.xml

The web.xml file contains a number of tags that define both servlets and their behavior. 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 the AJAX-Server 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. Various parameters within its tag define where to retrieve documents from, how it should render and deliver them to VirtualViewer, 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. 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>RetrievalServler</servlet-name>
<url-pattern>/ResponseServer</url-pattern>
 </servlet-mapping>
<servlet-mapping>
<servlet-name>UploadServler</servlet-name>
<url-pattern>/RequestServer</url-pattern>
</servlet-mapping>
```

Display Your Documents

If you want to change the location where you store your test images, the `filePath` parameter must be changed. If you want to store your images in the `C:/imgs` directory, then this parameter does not need to be edited. The `filePath` parameter tells the default content handler where to look for image files that are requested. Set the value to the path where you will put your test images as shown in the following location.

To view your own images in VirtualViewer AJAX, you must put the document that you want to view in the `C:\imgs` directory or the directory that the `filepath` parameter specifies in the web.xml file.

```
<param name="filePath" value="C:/imgs/"/>
```

You must then append the parameter `documentId` to the end of the URL in order to specify the ID of the document you want to display. For example, if you want to display the file named commerce.tif, add that name to after `documentId` as shown in the following example:

Example 3.4: Specifying the Document to Display

```
http://server:port-
/VirtualViewerJavaAJAXServer/ajaxClient.html?
documentId=filename.exthttp://server:port-
/VirtualViewerJavaAJAXServer/ajaxClient.html?
documentId=commerce.tif
```

If you are not able to get your images to load, please submit a support ticket at www.support.snowbound.com or see the "Please wait while your image is loaded" Message Displays Indefinitely topic in 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 AJAX through the config.js file. This file is included with your installation in the VirtualViewerJavaAJAX 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.

Integrated Mode

You can use the InitParams `contentServerType` to indicate that you want the AJAX server to serve as its own content server. This improves performance by eliminating the steps of encoding and transferring the document content from the content server to the Ajax server.

> **Note:**
> If you would like to use this enhancement, then please make sure that the web.xml file is configured as explained below.

In the web.xml file, for the `AjaxServlet`, indicate that you want the `AjaxServlet` to serve as its own content server.

Change the value of the `contentServerType` parameter from the default value of http to integrated:

Example 3.5: Changing the contentServerType parameter default value

```
<init-param>
```

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

Once the `AjaxServlet` is configured to run in integrated mode, you should then also make sure to update any relevant Content Server related init-params. Most importantly, this would include the Content Handler parameters such as the following:

Example 3.6: Updating the Content Server related init-params

```
<init-param>
 <param-name>contentHandlerClass</param-name>
 <param-value>com.snowbound.snapserv.servlet.FileContentHandler</param-
value>
</init-param>
<init-param>
  <param-name>filePath</param-name>
 <param-value>/Users/imgs/</param-value>
</init-param>
```

Customizing the User Interface

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

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

Example 3.7: Customizing What is Displayed in the AJAX Client

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

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

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

Turning On Buttons that are Off by Default

Some buttons may be turned off by default. For example, to turn the Flip Horizontal and Flip Vertical buttons back on, locate the names of the Flip Horizontal (flip_horiz.png) and Flip Vertical (flip_vert.png) buttons. They will most likely be in the following location: `<wwwroot>\VirtualViewerNetAjaxClient\resources\`

Next, add the following code into your ajaxClient.html file:

Example 3.8: Turning on Buttons that are Off by Default

```
<img id="flip_horiz"
class="widget"
src="resources/flip_horiz.png"
onclick="javascript:myFlexSnap.flipX()"
onmousedown="javascript:myFlexSnap.mouseDown('flip_horiz')"
onmouseup="javascript:myFlexSnap.mouseUp('flip_horiz')"
title="Flip Horizontally"
alt="Flip Horizontally" />
<img id="flip_horiz"
class="widget"
src="resources/flip_horiz.png"
onclick="javascript:myFlexSnap.flipX()"
onmousedown="javascript:myFlexSnap.mouseDown('flip_horiz')"
onmouseup="javascript:myFlexSnap.mouseUp('flip_horiz')"
title="Flip Horizontally"
alt="Flip Horizontally" />
<img id="flip_vert"
class="widget"
src="resources/flip_vert.png"
onclick="javascript:myFlexSnap.flipY()"
onmousedown="javascript:myFlexSnap.mouseDown('flip_vert')"
onmouseup="javascript:myFlexSnap.mouseUp('flip_vert')"
title="Flip Vertically"
alt="Flip Vertically" />
```

Configuring the Document Thumbnail Panel Display

You can set the `multipleDocMode` parameter in the config.js file to configure which documents will be shown within the Docs pane of VirtualViewer AJAX. 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 3.9: 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 3.10: 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:
> `ajaxClientDefault.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 3.11: Setting multipleDocMode to specifiedDocuments
This example shows how to set the `multipleDocMode` parameter in the config.js file to use `specifiedDocuments`.
`var multipleDocMode = multipleDocModes.specifiedDocuments;`
Documents are passed to the viewer via the configuration parameter: **`var SD`**
Add a new line to config.js defining var SD as shown in the following example:
`var SD = new Array("filename.type","filename.type","filename.type");`
Documents are loaded into the viewer with the onload event:
`<body onload="myFlexSnap.initViaURL()">`

Example 3.12: 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 ajaxClient.html file, change the value of the onload event.
Results in ajaxClient.html:
`<body onload="myFlexSnap.initSpecifiedDocuments(SD);">`

Hiding the Thumbnail Panel

The Thumbnail panel provides a convenient way to:

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

However, this convenience does have a price. VirtualViewer performance degrades because it is processing every page in the document Thumbs panel and/or the first page of every document in the Docs panel . If you want to speed up performance, you may want to disable or hide the thumbnail navigation panels 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. This disables the Page Manipulations menu in VirtualViewer 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 3.13: Configuring the Approved and Denied Rubber Stamps

```
var rubberStamp = [
{ textString: "Approved",
fontFace: "Times New Roman",
fontSize: 30,
fontBold: true,
fontItalic: true,
fontUnderline: 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

Displaying the Include Annotations Checkbox

Export Dialog Box: 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;
```

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 3.14: Turning on Redaction Support

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

Improving Performance or Quality

One of the differences between raster and vector formats is that raster formats have specific DPI 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 - AJAX Servlet web.xml Parameter.

Example 3.15: 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:

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.

Bit Depth Parameter Values and Description

Parameter Name	Description
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 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 - AJAX Servlet web.xml Parameter.

Example 3.16: Setting the DPI

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

Table 3.17: The available DPI parameters are shown in the table below:

DPI Parameter Values and Description

Parmeter 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 - AJAX Servlet web.xml Parameter.

Example 3.18: Setting the Format

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

Table 3.19: The available format parameters are shown in the table below:

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 C.

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 3.20: 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 Java and VirtualViewer Java AJAX* for more information.

Default Configuration Maximizes Performance

Please note that the default configuration for VirtualViewer 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 3.21: 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.

Configuring to Maximize Quality

Please note that the default configuration for VirtualViewer 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.

Clearing a Document from the Cache

You can improve performance by clearing documents from the cache when you save. To clear a document from the cache when you save, set the `clearCacheOnSave` parameter to true in your `web.xml` file in the `UploadServer` servlet section:

Example 3.22: Clearing a Document from the Cache

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

Defining the Number of Pages Cached in Memory

You can improve performance on the server by allowing VirtualViewer to retain pages in the cache. The less requests VirtualViewer makes to the server the better performance you will see on that server. However, this may impact performance on the client JRE as VirtualViewer is now building memory. It is important to find a balance between the two by performance tuning the cache size during testing. The `maxCachePages` applet parameter will define the number of pages VirtualViewer 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 3.23: Defining the Number of Pages Cached in Memory

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

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 Java AJAX

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:**
> Exporting virtual documents in original format is not supported for VirtualViewer AJAX. In version 1.8 of VirtualViewer AJAX, exporting virtual documents in original format is enabled and virtual documents are exported to a TIFF file using the AJAX Java server.

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.

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/VirtualViewerJavaAJAXServer/ajaxClient.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/VirtualViewerJavaAJAXServer/ajaxClient.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.

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 it 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

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.
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

Permission	Definition
	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, doc-
umentKey, annotationKey)
```

This method returns a hashtable 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 will pass both the `permissionLevel` and the `redactionFlag` to the `saveAnnotationContent` method in a Hashtable:

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.

FileNet Annotations

You can save Snowbound and FileNet annotations. You can also edit and delete existing FileNet annotations when the system is configured to save 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 AJAX servlet web.xml files as shown in the example below:

Example 4.5: 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:

```
var oneLayerPerAnnotation = true;
```

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

Example 4.6: Adding the annotationOutputFormat parameter Set to Snowbound

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

If VirtualViewer 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.

Annotation Mapping

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

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 comes with a file content handler that connects VirtualViewer 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.

What is the Content Handler?

The VirtualViewer content handler is a Java class that the serlvet will call on to perform various actions concerning the retrieval and storage of content. By default, the VirtualViewer 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 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 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 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.

Plugging in a Custom Content Handler

The VirtualViewer servlet will instantiate the content handler class that is specified in the application's web.xml using the parameter `contentHandlerClass`. For example:

Example 4.7: Setting Up the contentHandlerClass Parameter

```
<init-param>
      <param-name>contentHandlerClass</param-name>
```

```
        <param-value>com.snowbound.flexsnap.custom.MyContentHandler
</param-value>
</init-param>
```

FlexSnapSIContentHandlerInterface

This interface defines methods for retrieving content for VirtualViewer. 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:

Method	Type	Description
KEY_CLIENT_INSTANCE_ID	String	Value of the clientInstanceId applet 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.

Extracting Parameters from ContentHandlerInput

There are two methods with which you can extract parameters from the ContentHandlerInput hashtable.

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.

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 Hashtable 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 content.
getDocumentFile()	Returns the getdocumentFile parameter.

Method	Description
`getDocumentId()`	Returns `documentId` parameter.
`getHttpServletRequest()`	Returns a `HttpServletRequest` object.

The second method is by explicitly calling the get function on the input hashtable. 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 hashtable for extracting parameter data.

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` applet 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` hashtable 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);
```

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_PROP-ERTIES	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, "Secu-
rity 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.

FlexSnapSIContentHandlerInterface

Return Value	Method
ContentHandlerResult	deleteAnnotation(ContentHandlerInput input)

Return Value	Method
	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 hashtable.
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`.
ContentHandlerResult	`sendDocumentContent(ContentHandlerInput input)`

Return Value	Method
	This method gets called to send an image via a mechanism defined by the implementor.

FlexSnapSISaverInterface

`FlexSnapSIContentHandlerInterface` extends FlexSnapSISaverInterface	
ContentHandlerResult	`publishDocument(Con-tentHandlerInput input)`
ContentHandlerResult	`saveAnnotationContent (ContentHandlerInput input)`
ContentHandlerResult	`saveBookmarkContent (ContentHandlerInput input)`
ContentHandlerResult	`saveDocumentComponents (ContentHandlerInput input)`
ContentHandlerResult	`saveDocumentComponentsAs (ContentHandlerInput input)`
ContentHandlerResult	`saveDocumentContent (ContentHandlerInput input)`

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:

Method	Type	Description
KEY_CLIENT_INSTANCE_ID	String	Value of the clientInstanceId applet 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.

eventNotification

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

Implement this content handler method to receive event notifications.

Parameters

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

Method	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, and they will be named KEY_EVENT_SAVE_ANNOTATION_LAYER_NAME_BASE0 KEY_EVENT_SAVE_ANNOTATION_LAYER_NAME_BASE1 KEY_EVENT_SAVE_ANNOTATION_LAYER_NAME_BASE2
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'
VALUE_EVENT_EXPORT	String	The event being logged is a document export request.
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 (Con-
tentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request the content for the specified annotation key in the form of a byte array.

Example 4.8: 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:

Method	Type	Description
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet 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:

Method	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.

Example 4.9: 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:

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

Returns

A `ContentHandlerResult` object containing the following data:

Method	Type	Description
`KEY_ANNOTATION_NAMES`	`String`	The names of all annotation layers.

`getAnnotationProperties`

```
public ContentHandlerResult getAnnotationProperties (Con-
tentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to request the properties for a specified annotation layer in the form of a hashtable. For more information, see Annotation Security Overview.

Parameters

A `ContentHandlerInput` object containing the following data:

Method	Type	Description
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet 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:

Method	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 4.10: 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:

Method	Type	Description
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet parameter.

Returns

A `ContentHandlerResult` object containing the following data:

Method	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:

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

Returns

A `ContentHandlerResult` object containing the following data:

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

`getClientPreferencesXML`

```
public ContentHandlerResult getClientPreferencesXML (Con-
tentHandlerInput 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:

Method	Type	Description
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet parameter.

Returns

A `ContentHandlerResult` object containing the following data:

Method	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.

Example 4.11: 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:

Method	Type	Description
`KEY_HTTP_SERVLET_REQUEST`		The standard `HttpServletRequest` data.
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet parameter.
`KEY_DOCUMENT_ID`	`byte[]`	The contents of the document.

Returns

A `ContentHandlerResult` object containing the following data:

Method	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:

Method	Type	Description
KEY_CLIENT_INSTANCE_ID	String	Value of the clientInstanceId applet 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 (Con-
tentHandlerInput 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:

Method	Type	Description
KEY_CLIENT_INSTANCE_ID	String	Value of the clientInstanceId applet 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 (Con-
tentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

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

Parameters

A `ContentHandlerInput` object containing the following data:

Method	Type	Description
`KEY_HTTP_SERVLET_REQUEST`		The standard Java `HttpServletRequest` object.
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet 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 E.
`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:

Method	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 the applet will 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 in the applet.

Parameters

A `ContentHandlerInput` object containing the following data:

Method	Type	Description
`KEY_HTTP_SERVLET_REQUEST`		The standard Java `HttpServletRequest` object.
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet param-

Method	Type	Description
		eter.
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 E.
KEY_DOCUMENT_CONTENT	byte[]	The data of the document.

Returns

A ContentHandlerResult object containing the following data:

Method	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 the applet will 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 (Con-
tentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to save an annotation layer.

Parameters

A ContentHandlerInput object containing the following data:

Method	Type	Description
KEY_HTTP_SERVLET_REQUEST		The standard Java HttpServletRequest data.
KEY_CLIENT_INSTANCE_ID	String	Value of the clientInstanceId applet 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 (Con-
tentHandlerInput input)
throws com.snowbound.snapserv.servlet.FlexSnapSIAPIException
```

Called to save bookmark data.

Parameters

A `ContentHandlerInput` object containing the following data:

Method	Type	Description
`KEY_HTTP_SERVLET_REQUEST`		The standard Java `HttpServletRequest` object.
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet 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 (Con-
tentHandlerInput 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 applet.

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:

Method	Type	Description
`KEY_HTTP_SERVLET_REQUEST`		The standard Java `HttpServletRequest` object.
`KEY_CLIENT_INSTANCE_ID`	`String`	Value of the `clientInstanceId` applet 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 E.
`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 4.12: Calling saveAnnotationContent

```
AnnotationLayer[] ann = input.getAnnotationLayers();
for (int annIndex = 0; annIndex < annotations.length; annIndex++)
{
input.put(ContentHandlerInput.KEY_CLIENT_INSTANCE_ID, clien-
tInstanceId);
input.put(ContentHandlerInput.KEY_DOCUMENT_ID, documentId);
input.put(ContentHandlerInput.KEY_ANNOTATION_ID, ann[index].get-
LayerName());
input.put(ContentHandlerInput.KEY_ANNOTATION_CONTENT, ann[index].get-
Data());
input.put(ContentHandlerInput.KEY_ANNOTATION_PROPERTIES,
            ann[index].getProperties());
saveAnnotationContent(input);
}
```

Returns

A `ContentHandlerResult` object containing the following data:

Method	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(Con-
tentHandlerInput 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:

Method	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`	`AnnotationLayer[]`	The information for all annotation layers.
`KEY_BOOKMARK_CONTENT`	`byte[]`	The XML data for bookmarks.

Returns

A `ContentHandlerResult` object containing the following data:

Method	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:

Method	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:

Method	Type	Description
`DOCUMENT_ID_TO_RELOAD`	`String`	The `documentId` to load after a save is

Method	Type	Description
		made.

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

Appendix A - Config.js Parameters

This appendix lists and describes the parameters found in config.js. You can use the config.js file to configure the appearance of VirtualViewer AJAX. It allows you to configure colors, zoom levels, multiple documents mode, and error messages. This file is included with your installation in the VirtualViewerJavaAJAX directory.

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;
```

Descriptions of Config.js Parameters

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

Supported Config.js Parameter Descriptions

Name	Default	Description
`servletPath`	"/VirtualViewerJavaAJAXServer/AjaxServlet"	Specifies the path to the AJAX servlet.
`vvHelpPath`	"resources/WebHelp/virtualviewer.htm";	Specifies the path to the online manual.
`maxZoomPercent`	1000	Sets the percentage to stop allowing users to zoom the image.
`zoomTimeout`	300	Sets the wait in X milliseconds before requesting the zoomed 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 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 false, does not include annotations when `sendDocument` is called.
`errorColorStrings`	"Invalid color string."	Sets the error message displayed when the user inputs an invalid color string.

Name	Default	Description
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, retain-ViewOptionsBetweenPages should ensure that the viewer does not go back to the defaultZoomMode setting when switching between pages.
defaultZoomMode	zoom-Modes.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. 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 fitLastBetweenDocuments 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 should burn the annotations into the image when printing.
exportBurnAnnotations	false	Determines if VirtualViewer should burn the annotations into the image when exporting.
oneLayerPerAnnotation	false	Create a new annotation layer for each annotation.

Name	Default	Description
`helpURL`	"help/help.html";	Passed to window.open when creating the help window. `window.open(helpURL,helpWindowName,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,helpWindowName,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(helpURL,helpWindowName,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.

`multipleDocMode`	`multipleDocModes.availableDocuments`	Sets the multiple documents mode. You can use any of the following variables:
		`availableDocuments`
		`viewedDocuments`
		`specifiedDocuments`
		Please see Configuring the Document Thumbnail Panel Display for more information on configuring the multipleDocMode parameter.

`pageManipulations`	true	Sets the ability to use the Page Manipulation functionality.
`enableRubberStamp`	true	When set to true, enables the Rubber Stamp functionality. When set to false, disables it.

Name	Default	Description
RubberStamp	{ textString: "Approved", fontFace: "Times New Roman", fontSize: 30, fontBold: true, fontItalic: true, fontUnderline: true, fontColor: "00FF00" }, { textString: "Denied", fontColor: "FF0000" }];	Configure the two Rubber Stamps **Approved** and **Denied** by default.
errorDeleteLayerPermissionString	"You do not have permission to delete this layer."	Sets the error message displayed when the user tries to deletes a layer permission string.
errorRenameLayerPermissionString	"You do not have permission to rename this layer."	Sets the error message displayed when the user tries to rename a layer permission string.
errorCreateAnnLayerPermissionString	"You do not have permission to create annotations on this layer."	Sets the error message displayed when the user tries to create annotations on a layer.
errorEditAnnLayerPermissionString	"You do not have permission to edit annotations on this layer."	Sets the error message displayed when the user tries to edit annotations on a layer.
errorLayerInvalidNameString	"Invalid layer name. Please choose a new layer name:"	Sets the error message displayed when the user chooses an invalid layer name.
errorLayerNameExistsString	"A layer with that name already exists."	Sets the error message displayed when the user selects a layer that already exists.
errorTabTooManyTabs	"Too many open tabs."	Sets the error message displayed when the user tries to open more tabs

Name	Default	Description
		then allowed.
errorTabIndexOutOfBounds	“Tab index out of bounds."	Sets the error message displayed when the tab index is out of bounds.
errorTabCloseLastTab	"Closing last tab is not allowed."	Sets the error message displayed when the closing tab is not allowed.
vvStatusSavingDocument	"Please wait while your changes are saved."	Sets the message displayed when saving a document.
vvStatusWaitIndicator	"Please wait while your image is loaded."	Sets the message displayed while and image is loading.
vvDele-teAnnotationDialogTitleString	"Delete annotation?"	Sets the message displayed when deleting an annotation.
vvDele-teAnnotationDialogTextString	"Are you sure you wish to delete this annotation?"	Sets the message displayed to confirm deleting an annotation.
vvE-dit-TextAnnotationDialogTitleStrin	"Edit Annotation Text?"	Sets the message displayed when editing annotation text.
vvE-dit-TextAnnotationDialogTextString	"Annotation Text"	Sets the messaged displayed for annotation text.

Please see the next topic Appendix B - AJAX Servlet web.xml Parameter.

Appendix B - AJAX Servlet web.xml Parameters

This appendix lists and describes the AjaxServlet web.xml parameter.

The web.xml file contains a number of tags that define both servlets and their behavior. 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 the AJAX-Server web.xml when the `contentServerType` parameter is set to integrated. For more information, please see Configuring web.xml.

Description of the AJAX Servlet Parameters

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

Supported AJAX Servlet Parameters

Name	Default	Description
`annotationOutputFormat`	snow-bound	Enables the ability to edit and delete existing FileNet annotations. When set to FileNet, annotations are saved in FileNet XML format. When set to Snowbound, annotations are saved in Snowbound XML format.
`contentServerType`	http	If set to integrated, the VirtualViewer Java AJAX servlet will work as its own content server.

Please see the next topic Appendix C - Servlet Tags for web.xml.

Appendix C - Servlet Tags for web.xml

This appendix lists and describes all servlet tags for web.xml

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

The AJAX Servlet 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 C.1: Setting contentServerType to Integrated

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

For more information, please see Appendix B, AJAX Servlet web.xml Parameter.

The appendix contains the following topics:

ReponseServer Servlet Parameters

Required Servlet Parameters

Optional Servlet Parameters

UploadServlet Servlet Parameters

Deprecated Servlet Parameters

Obsolete Servlet Parameters

ResponseServer Servlet Parameters

This table lists and describes the ResponseServer servlet parameters.

ResponseServer

Name	Default	Description
`filePath`	`C:\imgs`	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.

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.

RetrievalServlet

Name	Default	Description
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.
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.
documentCacheSize	400000	The size in bytes of the server document cache.
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.
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 snbd_map.fnt file

Name	Default	Description
iocaBitDepth	1	The bit depth to use when decompressing IOCA pages. Valid values are 1 or 24.
iocaDPI	200	The Dots Per Inch 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.
jpegQuality	50	Level of quality when a page is converted to JPEG and sent to VirtualViewer.
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
modcaDPI	200	The Dots Per Inch 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.
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
pclDPI	200	The Dots Per Inch 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
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

<table>
<tr><th>Name</th><th>Default</th><th>Description</th></tr>
<tr><td>pptDPI</td><td>200</td><td>The dots per inch to use when decompressing PPT pages.</td></tr>
<tr><td>pptFormat</td><td>JPEG</td><td>The format to convert PPT pages to. Valid values are TIFF_G4_FAX, JPEG, TIFF_LZW, PNG</td></tr>
<tr><td>preferencesPath</td><td>N/A</td><td>Specifies the location of stored client preferences on the server when using the default Content Handler.</td></tr>
<tr><td>supportRedactions</td><td>false</td><td>Turn on redaction support.</td></tr>
<tr><td>thumbByteEstimate</td><td>6000</td><td>The initial byte size of the buffer used on the server to transport thumbnails.</td></tr>
<tr><td>thumbnailDPI</td><td>60</td><td>Specifies the DPI to use when rendering thumbnails for vector formats such as PDF and MS Word.</td></tr>
<tr><td>tiffByteSize</td><td>40000</td><td>Initial size of the byte array when saving to TIFF to send to VirtualViewer.</td></tr>
<tr><td>vectorPDF</td><td>false</td><td>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:

<table><tr><th>Server</th><th>Client</th><th>Result</th></tr><tr><td>true</td><td>true</td><td>Client draws the PDF.</td></tr><tr><td>true</td><td>false</td><td>Client renders the PDF.</td></tr><tr><td>false</td><td>either</td><td>Server renders the PDF.</td></tr></table></td></tr>
<tr><td>wordBitDepth</td><td>24</td><td>The bit depth to use when decompressing Word pages. Valid values are 1 or 24.</td></tr>
<tr><td>wordDPI</td><td>200</td><td>The dots per inch to use when decompressing Word pages.</td></tr>
<tr><td>wordFormat</td><td>JPEG</td><td>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.</td></tr>
<tr><td>xlsBitDepth</td><td>24</td><td>The bit depth to use when decompressing XLS</td></tr>
</table>

Name	Default	Description
		pages. Valid values are 1 or 24.
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 Servlet Parameters

This table lists and describes the UploadServlet servlet 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 over-ridden by the equivalent applet parameter.
emailServer	N/A	Sets the default SMTP Server for emails sent via Email Document. This can be over-ridden by the equivalent applet 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 PDF.format=PDF JPEG.format=JPEG default.format=TIFF_LZW
permanentAnnotationLinks	true	WEBTOP VERSION ONLY. If false, keeps

Name	Default	Description
		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

This table lists and describes the ResponseServer parameters.

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.
pngForPDF	false	Specifies that PDF pages should be converted to PNG. (replaced by convertPDF, and then by pdfFormat)

Obsolete Servlet Parameters

This table lists and describes the ResponseServer parameters.

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.

Please see the next topic Appendix D - JavaScript APIs.

Appendix D - JavaScript API

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

JavaScript API

This table lists and describes all JavaScript API

Supported JavaScript API Descriptions

Name	Returns	Description
`getPageNumber`	`String`	Returns a String representing the page number of the page currently being viewed.
`set-ClientInstanceId`	`Void`	Allows the `clientInstanceId` to be set via Java-script method.

Please see the next topic Appendix E - Supported File Formats.

Appendix E - Supported File Formats

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

VirtualViewer 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 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.

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 www.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

This table lists and describes all supported file formats.

Supported File Format Descriptions

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
AFP (MO:DCA) *	74	1	1	See MO:DCA. his is a multi-page file format.
ASCII	38	1	No	Snowbound reads in ASCII text files and converts them to a bitmap.
BMP_COM-PRESSED	12	4, 8	No	Originated by Microsoft, BMP supports 1, 4, 8, and 24-bit images.
BMP_UNCOM-PRESSED	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

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				.PCX file in whole which can be 1, 4, 8, and 24-bit.
DIB	48	1, 4, 8, 24	No	Standard Windows Device 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.
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/servlet/item?siteID=123112&id=8446698
EMAIL	89	1	1	E-mail message created with MS

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				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, 321	EPS Compressed bitmap format. It is an Adobe encapsulated Postscript file with either G4 or JPEG data embedded.
EPS_BITMAP_G4	64	No	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	No	1, 8, 24, 321, 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.
GX2	22	4, 8	No	Originated by Brightbill Roberts for ShowPartner DOS applications. Supports 4 and 8-bit images. Simple run length encoding technique.

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
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 bit-map. Supports 1 and 4 bits uncom-pressed.
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	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 for-mat.
JBIG *	71	1	1 (with plu-gin) **	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	No1 (with plu-gin) **	JBIG2 is a highly-compressed black and white image format that uses symbol recognition and sub-stitution for very dramatic com-pression results. Snowbound's viewers and conversion programs can be used to directly view JBIG2 documents or convert those doc-uments to a variety of output for-mats.
JEDMICS *	56	1	1	US Military CCITT G4 tiled image format for storing Government doc-uments and drawings. Supports 1-bit per pixel.
JPEG	13	8, 24, 32	8, 24, 32	Joint Photographics Experts Group. This was a group spear-

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				headed 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	8, 24	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	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, presentations and word processing documents.
ODP	101	No	No	Open Document Format for presentations.
ODS	97	No	No	Open Document Format for spread-

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				sheets.
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 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

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				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 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 for-

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				mat 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 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

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				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.
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.
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 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

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
				Aspose.Total.Family.P-roduct.License (eval license) dom4j-1.6.1.jar log4j-1.2.16.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 Cor-poration. Gray scale color and CMYK color images. Usually com-pressed.
TARGA	3	8, 16, 24, 32	8, 16, 24, 32	The SCITEX format is a proprietary format originated from SCITEX Cor-poration.
TARGA16	3	8, 16, 24, 32	8, 16, 24, 32	The SCITEX format is a proprietary format originated from SCITEX Cor-poration.
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 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.

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
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 G, "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.
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, 8	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.

File Format	File Type Number	Input Bit Depth	Output Bit Depth	Description
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.
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

This table lists the available 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
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

File Type Number	File Type Name
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
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

File Type Number	File Type Name
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

Please see the next topic Appendix F - Snowbound Error Codes.

Appendix F - 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.

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 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 www.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.
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 E, 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.
DLL_NOT_LOADED	-24	DLL not loaded for Win 3.x version. There was an error loading a DLL. Please open a support issue at www.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.
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

Error	Error Code	Description
		file format. Please open a support issue at www.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 www.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.
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 Snowbound library version.
ACCESS_DENIED	-46	Access denied. Please check the security permissions.

General Error Define Values from Status Property

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

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.IMAGE_NOT_AVAILABLE	-103	Internal image data unavailable when trying to 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.

General Status/Error Codes

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.

Error	Description
SYSTEM_CRASH	This is returned when a Critical Exception is thrown.

Appendix G - Troubleshooting

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

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

In some cases, images do not load in the VirtualViewer AJAX 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 AJAX 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 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 JAVA AJAX, 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

AJAX .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="AJAXServer" type="Snowbound.VirtualViewerNetAJAXServer.AjaxServerHandler, Snowbound.VirtualViewerNetAJAXServer" />

</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 AJAX Java, check the `contentHandlerClass` parameter value. For VirtualViewer AJAX .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 following 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 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`.

Default Configuration Maximizes Performance

Please note that the default configuration for VirtualViewer 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 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 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.

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*.

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 - config.js.
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.

Index

4

404 Not Found 111
405 Method Not Allowed 111

5

500 Internal Server error 112

A

ACCESS_DENIED 108
AFP(MO:DCA) 91
AjaxServlet 13
annotation
  text does not appear 113
annotation key 54
annotation Mapping 56
annotation security 52
  level definitions 53
  permission levels 52
annotationOutputFormat 56, 82
annotations 18
  creating 18
  delete 19
  deleting 19
  edit text 19
  exporting 24
  FileNet 56
  moving 19
  print 25
  properties 19
  resizing 19
  rubber band stamp 20
  saving 19
  Snowbound 56
  undo delete 20
Apache Tomcat 10
ASCII 90-91
  filter bit level support 90
AUTOFED_FAILED 108
availableDocuments 34, 40

B

BAD_DISPLAY_AREA 106
BAD_HANDLE 107
BAD_RETURN 106
BAD_STRING 106
baseURL 84
BEA Weblogic 8.1 10
bit depth
  parameters 45
  setting 45
bitDepth 84
BMP_COMPRESSED 91
BMP_UNCOMPRESSED 91
BROOK_TROUT 91
browser
  running VirtualViewer 15
browsers 10
burned in 53
buttons
  turning on 40

C

cache
  pages to memory 50
CacheValidator 58

CALS 91
CANT_CREATE_FILE 106
CANT_FIND_TWAIN_DLL 107
CCITT_G3 91
CCITT_G3_FO 91
CCITT_G4 91
CCITT_G4_FO 91
CFF 91
CIMS(ABIC) 91
clearCacheOnSave 87
CLIP 91
COD 91
codebase 14
COMPRESSION_NOT_SUPPORTED 107
concurrentBWThumbs 88
concurrentColorThumbs 88
concurrentImages 88
config.js 15
configuring
  config.js 15
  thumbnail panel 40
  web.xml 14
content handler 17, 57
  custom 57
Content Handler methods 61
content server 10
  integrated 13
ContentHanderResult 58
contentHandlerClass 39, 84
ContentHandlerInput
  extracting parameters 59
  populating 60
ContentHandlerResult
  populating 60
ContentHandlerResult.ERROR_MESSAGE 61
contentserverType 13
contentServerType 38, 82
convertPDF 88
Copy to New Document 31
  disabling 43
copying pages 31
CORRUPTED_FILE 106
customizing
  display 39
  environment 14
  location of documents 14
  VirtualViewer 36
CUT 91
cutting pages 31

D

DCS 91
DCX 91
defaultByteSize 84
delete
  annotations 19
  layer 23
DELETE_ERROR 109
deleteAnnotation 63
deleting
  annotations 19
deleting pages 31
DIB 92
DICOM 92
DISK_FULL 106
DISK_READ_ERROR 106
DISPLAY_ERROR 109

DLL_NOT_LOADED 107
DOC 92
document
  sending 25
document store
  connecting 56
documentCacheCount 88
documentCacheSize 84
documents
  customizing location 14
  displaying 37
DOCX 92
docxBitDepth 84
docxDPI 84
docxFormat 84
DPI
  parameters 46
  setting 46
DWG 92
DXF 92

E

edit text
  annotations 19
EMAIL 92
emailFromAddress 87
emailServer 87
enableRubberStamp 20, 43
environment
  customizing 14
EPS 93
EPS_BITMAP 93
EPS_BITMAP_G4 93
EPS_BITMAP_IZW 93
error
  display in client 61
  return for display in client 61
ERROR_OPENING_SCANNER 107
errorColorStrings 77
errorCreateAnnLayerPermissionString 80
errorRenameLayerPermissionString 80
errorTabIndexOutOfBounds 81
errorTabTooManyTabs 80
EVAL_TIMEOUT 107
evaluation version 11
eventNotification 64
EXCEPTION_ERROR 108
exceptions
  supported file formats 10
exportBurnAnnotations 44
exporting document
  annotations 24
extractJpeg 84
extractPDFPages 84
extractTiffJpeg 84

F

FEEDER_NOT_READY 107
file formats
  exceptions to supported 10
FILE_NOT_FOUND 106
FileNet
  annotations 56
FILENET 93
filePath 16, 37, 83
Firefox 10

fit-to-page 27
FLASHPIX 93
FlexSnapSIAPIException 61
FlexSnapSIContentHanderInterface 58
FlexSnapSIContentHandlerInterface 61
fontMappingPath 84
fonts
  do not display 113
format 38
  setting 46
format parameters
  setting 38
FORMAT_NOT_ALLOWED 106
FORMAT_WILL_NOT_OTFLY 107

G

GENERAL_STATUS.DEFAULT 109
GENERAL_STATUS.DELETE_ERROR 109
GENERAL_STATUS.DISPLAY_ERROR 109
GENERAL_STATUS.IMAGE_NOT_AVAILABLE 109
GENERAL_STATUS.NOT_VALID 109
GENERAL_STATUS.SNOWBND_API_NOT_AVAILABLE 109
GENERAL_STATUS.SNOWBND_ERROR 109
GENERAL_STATUS.SNOWBND_OK 109
GENERAL_STATUS.SYSTEM_CRASH 109
getAnnotationContent 65
getAnnotationNames 65
getAnnotationProperties 66
getAnnotationPropertis 55
getAvailableDocumentIds 67
getBookmarkContent 67
getClientPreferencesXML 68
getDocumentContent' 68
getPageNumber 89
GIF 93
GIF_INTERLACED 93
GX2 93

H

hasAnnotations 69
helpURL 79
helpWindowName 79
hiding
  thumnail panel 28, 42
HTML 94

I

IBM Websphere 5.1 10
ICONTYPE 94
IFF_ILBM 94
IIS5 112
IMAGE_NOT_AVAILABLE 109
IMG 94
IMNET 94
inserting pages 31
installation 10
integrated content server 13
integrated mode 13

Internet Explorer 10
inverting 27
IOCA(MO:DCA) 94
iocaBitDepth 85
iocaDPI 85
iocaFormat 85
iPad 19
iPhone 19

J

J2EE 10
J2SE 10
Java version 10
JBIG 94
JBIG2 94
JEDMICS 94
JPEG 94
JPEG2000 95
jpegByteSize 85
jpegCompression 88
jpegQuality 85
JRE 10

K

key/value pairs 54
KOFAX 95

L

landscape
  display 117
LASER_DATA 95
layer
  deleting 23
  redact 23
  rename 23
layer manager 21
layers
  printing 55
LINE_DATA 95
location
  customizing documents 14
logLevel 85

M

MACPAINT 95
MAG 95
maxByteMultiplier 88
memory
  page cached 50
  recommended settings 115
METHOD_NOT_FOUND 108
MODCA_IOCA 95
modcaBitDepth 85
modcaDPI 85
modcaFormat 85
moving
  annotations 19
MSG 95
MSP 95
mulitple documents
  open 34
multipleDocMode 34, 79

N

NCR 95
new layer
  creating 23
NO_ABIC_VERSION 108

NO_BITMAP_FOUND 106
NO_CLIPBOARD_IMAGE 107
NO_DELAY_TIME_FOUND 107
NO_FAST_TWAIN_
SUPPORTED 108
NO_JPEG2000_VERSION 108
NO_LZW_VERSION 107
NO_MORE_PAGES 107
NO_PCL_VERSION 108
NO_PDF_VERSION 108
NO_SCANNER_FOUND 107
NO_TCOLOR_FOUND 107
NO_VECTOR_CAPABILITY 108
NO_WORD_VERSION 108
NOT_A_TILED_IMAGE 107
NOT_SUPPORTED_IN_THIS_VER-
SION 107
NOT_VALID 109

O

ODF 95
ODP 95
ODS 95-96
oneLayerPerAnnotation 55
OOXML 96
OUT_OF_MEMORY 106
outputConfigPath 87
overlayPath 85

P

packaging 10
page
selecting 29
page controls 27
page manipulation 27
loading context menu 30
page manipulations
overview 29, 43
saving 31
PAGE_NOT_FOUND 106
page-
Manip-
ulationsNewDocumentMenu 43
pages
cached to memory 50
copying 31
cutting 31
deleting 31
inserting 31
PALETTE_IMAGES_NOT_
ALLOWED 107
parameters
setting 46
PASSWORD_PROTECTED_
PDF\ 108
PCL 96
PCL_1 96
PCL_5 96
pclBitDepth 85
pclDPI 85
pclFormat 85
PCX 96
PDF 97
filter bit level support 90
PDF_15 97
PDF_16 97
pdfBitDepth 85
pdfDPI 85
pdfFormat 85

performance
maximizing 48
setting bit depth 49
setting DPI 49
permanentAnnotationLinks 87
permission levels 52
PhotoCD 98
Photoshop 98
PICT 98
PIXEL_DEPTH_
UNSUPPORTED 90, 107
pixelLimit 85
Please wait while your image
loads 111
PNG 99
pngForPDF 88
PPT 99
pptBitDepth 85
pptDPI 86
pptFormat 86
PPTX 99
preferencesPath 86
print 25
with or without annotations 25
printBurnAnnotations 44
printing
layers 55
virtual document 52
production version 11
properties
annotations 19
publishDocument 71

Q

quality
maximizing 49
setting bit depth 50
setting DPI 50

R

RAST 100
Read/Write Capabilities 90
redact
layer 23
redaction
turning on 44
redaction layers
saving 55
redactions 52
rename
layer 23
Request Server 36
resizing
annotations 19
Response Server 36
retainViewOptionsBetweenPages 78
retrieval servler 36
RTF 100
rubber band stamp
annotations 20
rubber band zoom 26
RubberStamp 80

S

Safari 10
Save 61

saveAnnotationContent 55, 72
saveAnnotationsAsXml 88
saveBookmarkContent 72
saveClientPreferencesXML 70
saveDocumentComponents 73
saveDocumentComponentsAs 74
saving
  annotations 19
  page manipulations 31
  redaction layers 55
SCITEX 100
SEARCH_STRING_NOT_
    FOUND 108
Send 61
sendDocumentWithAnnotations 77
sending
  document 25
servlet container 10
servlet paths 37
servletPath 77
servletURL 15
sessionClass 88
setClientInstanceId 89
SNOWBND_API_NOT_
    AVAILABLE 109
SNOWBND_ERROR 109
SNOWBND_OK 109
Snowbound annotations 56
specifiedDocuments 34, 41
support issue 117
supported
  exceptions to file formats 10
supportRedactions 44, 86
system requirements 10
SYSTEM_CRASH 110

T

TARGA 100
TARGA16 100
thumbByteEstimate 86
thumbnail and docs panel 27
thumbnail panel
  configuring 40
  hiding 28, 42
thumbnailDPI 86
TIFF LZW 101
TIFF UNCOMPRESSED 101
TIFF_2D 100
TIFF_ABIC 100
TIFF_ABIC_BW 101
TIFF_G3_FAX 101
TIFF_G4_FAX 101
TIFF_G4_FAX_FO 101
TIFF_G4_FAX_STRIP 101
TIFF_HUFFMAN 101
TIFF_JBIG 101
TIFF_JPEG 101
TIFF_JPEG7 101
TIFF_PACK 101
TIFF_TAG_NOT_FOUND 107
tiffByteSize 86
tmpDir 84

U

undo deleted
  annotations 20
Upload Server 36
USER_CANCEL 107
USING_RUNTIME 107

V

validateCache 58
vectorPDF 86
viewedDocuments 34, 40
virtual document
  printing 52
virtual documents 51
  loading 51
VirtualViewerJavaAJAX.war 12
VirtualViewerJavaAJAX.zip 12
vvDeleteAnnotationDialogTitleString 81
vvEditTextAnnotationDialogTextString 81
vvStatusSavingDocument 81

W

watermarks 52
WBMP 102
WEB-INF 14
web application 12
web.xml 14
  configuring 10
webapps directory 12
WINFAX 102
WMF 102
wordBitDepth 86
wordDPI 86
wordFormat 86
WPG 102

X

XBM 102
Xerox_EPS 102
XLS 93
xlsBitDepth 86
xlsDPI 87
xlsFormat 87
XLSX 102
XPM 102
XWD 102
  filter bit level support 90

Z

zooming 26