VirtualViewer®

V4.16 VirtualViewer® HTML5 for .NET Administrator's Guide

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

DOC.3 -VV .NET 4.16

Copyright Information

Manual Title: *Snowbound Software VirtualViewer® HTML5 for .NET Administrator's Guide*
Part No.: DOC-VV .NET 4.16
Revision: 1
VirtualViewer® HTML5 for .NET Release Number: 4.16
Published Date: June 2020

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

Comments about documentation: documentation@snowbound.com

Table of Contents

About Snowbound Software

For over two decades, Snowbound Software has been the independent leader in document viewing and conversion technology. It plays an integral role in enhancing and speeding document processing for the Fortune 2000. Snowbound excels in providing customers with powerful solutions for capturing, viewing, processing, and archiving hundreds of different document and image types. Thanks to its pure HTML5 technology and multi-environment support (including Java and Windows), Snowbound's products operate across all popular platforms and can be easily integrated into new or existing enterprise content management systems. Nine of the 10 largest banks in the United States (seven of 10 in the world), as well as some of the biggest healthcare providers, government agencies, and insurance companies rely on Snowbound for their mission-critical needs.

Important Phone Numbers and Links

For the most current information, please contact Snowbound Sales at:

1-617-607-2010

or

http://register.snowbound.com/MQL-contactUs-Website-2017.html

or

questions@snowbound.com

or

https://mylivechat.com/chatnoscript.aspx?HCCID=17729140 (sales inquiries only)

Release Notes and Product Manuals:

http://www.snowbound.com/support/manuals

Comments about documentation:

documentation@snowbound.com

Snowbound Target Industries

Snowbound's two flagship products—VirtualViewer® HTML5 (a pure HTML5 document viewer) and RasterMaster® SDK (document/image conversion library)—help organizations and companies across a variety of industries meet their document viewing and conversion needs:

- Medical: Patient record management
- Insurance: Insurance & health insurance claim processing
- Finance: Mortgage processing & financial statements
- Shipping: Full array of shipping documents
- Legal: Claims, briefs, and other court documents

VirtualViewer® HTML5

Easy-to-Use in Any Environment

VirtualViewer® HTML5 is equipped with powerful and sophisticated features and functionality.

True cross-platform support: VirtualViewer® HTML5 is a universal viewer that operates seamlessly on any platform with both a pure Java solution with Java-based server components or a .NET solution.

No Downloads: No application download or client-side installation is required, making it a trouble-free solution for users as well as IT administrators.

Localized UI: The viewer's intelligent localization capabilities auto-detect browser settings and display in the proper language.

High-speed viewing: With advanced server processing, the viewer delivers an extremely high-speed response.

Seamless Integration into ECM Applications: VirtualViewer® HTML5 integrates into existing back end repositories and homegrown applications. Snowbound also offers a variety of out of the box ECM connectors (Alfresco, IBM FileNet, and Open Text/Documentum) with seamless integration.

One Quick & Easy 10 Minute Installation

Installation of VirtualViewer® HTML5 takes less than 10 minutes for POCs on any desktop, laptop, and virtual machine. After the quick and easy install, VirtualViewer® HTML5 is then backed by Snowbound's award-winning and responsive support team. Snowbound's skilled network of system integrators can further enhance the benefits of VirtualViewer® HTML5 with custom integration to your existing system.

Technical Information

Snowbound provides the option of either a 100% Java or a .NET (64-bit) server component. The viewer operates in all modern browsers (Microsoft Edge, Firefox, Chrome, Safari, Microsoft Internet Explorer 11 and mobile browsers).

Server options:
- UNIX servers including Linux, Sun, IBM, HP, Mac
- Windows servers including Server 2016, 2012, 2010, 2008 and 2007. Server 2019 coming soon.

RasterMaster® SDK

RasterMaster® is the industry's leading document/image conversion and imaging library for Java and .NET. It is continually enhanced with new functionality and formats and was developed by Snowbound's experts who have nearly a hundred years of combined imaging expertise.

High-Speed File Conversion

RasterMaster® is the fastest file conversion SDK on the market. Users can quickly convert files on the fly for viewing or batch convert large amounts of document types. Special features, including conversion via Byte Array is also available for high performance applications.

Extensive Format Support

AFP, DWG, JPEG, MO:DCA, PDF, MS Office, TIFF, SVG, PNG, and hundreds more document types are supported. Convert any format to PDF or TIFF to ensure universal compatibility. RasterMaster® also includes both PDF/A and SVG output support, enabling long term archiving and high resolution viewing.

- **Technical Information**:
  RasterMaster® is available for multiple platforms, including Java and .NET:
- Java: for all computing platforms, including Unix, Linux, Windows, and Mac|
- NET (x64): for Windows native applications, including Server 2016, 2012, 2010, 2008, and 2007|

Responsive Support

All of Snowbound's products are backed by responsive support. Our expert, responsive internal support team is available to answer your questions and help you install our HTML5 viewer and conversion SDK. A support portal is also available 24x7 for questions and information at https://snowboundsupport.force.com/SupportPortal/CommunityLogin.

Important Information

For the latest information, please refer to the Release Notes (releasenotes.md) in your product shipment directory. However the latest versions will always be In the Release Notes version on the Snowbound website:

http://www.snowbound.com/support/manuals

- Please be advised the previously named “default content handler” and now called the “sample content handler” is actually intended to be used only for Proof of Concept efforts but is not a complete connector.|
- It is recommended that customers upgrade as soon as possible to the latest release of VirtualViewer (typically offered quarterly). The product is rapidly evolving with new features as well as fixes.|
- Snowbound recommends the use of the SVG output format from the server to the browser whenever possible for reducing data size and improving performance, particularly when working with large spreadsheets.|
- When working with large spreadsheets, it may be advantageous to try the file breakup option so you’re not working with extremely large downloaded documents that might affect performance.|
- VirtualViewer for Java now supports JRE 1.7 and 1.8. Previous JRE versions are no longer supported or tested except under special arrangements. It is expected that support will shift to JRE 1.8 in 2019.|
- For Windows products, .NET framework versions 4.5.2 and up are now supported|
- Web.xml changes: The following parameters in web.xml have been removed:|
  - defaultByteSize
  - tiffByteSize
  - jpegByteSize

Major Past Version Features in VirtualViewer®

v4.11

Documentation Corrections for VirtualViewer 4.11

- VirtualViewer for Java now supports JRE 1.7 through December 2018. Previous JRE versions are no longer tested or supported. Note that Oracle has accelerated Java releases and we encourage our customers to follow that model in order to insure security of your applications.|
- For Windows products, .NET framework versions 4.5.2 and up are now supported|

Document Compare

This feature will take the text of two documents open in the viewer and compare them together. The results of this comparison are displayed in a new tab in the right-hand thumbnail pane, and users can navigate through each edit to the document.

Callback Event Manager

Starting with 4.11, there is a single API method for setting callbacks and a new manager for the callbacks.

Sticky Note Background Color Support

User Preference Option for Default Thumbnail Tab

Copy Annotations Across Documents

Get Page Dimensions API call

Require.js

We now use Require.js to compile and load our javascript code.

InitSpecifiedDocuments and OpenSpecifiedDocuments

The `initSpecifiedDocuments` API now allows more control over how VirtualViewer opens.

Toggle Annotation Visibility

This new API allows toggling visibility of all annotations on a document.

v4.10

Mobile Device Control Improvements

- The user can pan on an image, if it's zoomed in, in all directions. Using two fingers. the user can pinch to zoom in or zoom out on the document.|
- The viewer now works more elegantly in small iframes, on low-resolution monitors, and when browser windows resized smaller.|

Alfresco Quickshare Support

A logged-in user can create or close a public quickshare link through Alfresco.

Alfresco Watermark Support

Added support for Watermarks in the Alfresco version of VirtualViewer allowing saving watermarks back into the Alfresco repository.

Revision history for Annotation Create Date/Time

Filenet F_CREATOR Tag Support Added

NET 4.5.2+ Requirement Added for TLS 1.2 support

OCR Integration (beta)

v4.9

Watermarks

VirtualViewer® HTML5 now offers watermarks for users who need to mark page backgrounds with specific notifications such as "Private", "Confidential", and "Do Not Distribute." Users can easily add watermarks to their document via a new button on the left sidebar of the viewer. Additional watermarks functionality includes:

- The ability to customize the appearance of the watermark (direction, location, and sizing), the text of the watermark, and the opacity of the watermark (transparent or solid).|
- To expedite the process, the viewer provides the user with predefined watermarks such as "Edited by"; "time/date printed"; "page number"; "total pages"; and "document name."|
- Administrators can restrict who has access to the watermarks feature based on user permissions.|

Footers for Page and Document Thumbnails display filenames

Magnifier window resize uinsg the mouse as added

Notes Tab now toggles on if a document note is present

Document Notes templates capability has been added

OCR integration (beta)

v4.7

DWG Layer & xref Support
Users can easily access the DWG files created by CAD applications such as drawings and blueprints, as well as interact with the layers within those files individually. The user can decide which of those layers to view and which to take out of view, allowing for a streamlined review process where the user is only seeing the information they need and nothing else.

Split Screen View
The split screen view allows users to launch a lower panel to simultaneously compare documents side-by-side during the review process so they can easily spot differences, as well as display data in one view and manipulate in another. The user no longer needs additional tabs or windows to view multiple pages or documents at once. The result is a cleaner user experience, a streamlined review process, and less memory required on the server.

Extract and Append Page Ranges
The viewer can extract a range of pages instead of the entire document when saving to PDF, meaning users can save updated, large PDF documents at least 10 times faster than before. This results in dramatic speed improvements (in one instance, 215 seconds before and 3 seconds after) for users with many large, multi-page PDFs.

v4.6

Faster Performance
Snowbound has made upgrades which allow users to view, convert, and manage Microsoft Office documents - including Microsoft Word, Excel, and PowerPoint - at increased speeds. Benchmark testing showed that the viewer now loads these documents six or more times faster than before. These enhancements are also available in the new version of the firm's document conversion SDK, RasterMaster.

Pattern-Based Text Searching
Snowbound has the ability for users to search for patterns in text, including social security numbers, phone numbers, credit card numbers, and e-mail addresses. Users can use this feature to quickly locate, redact, or collaborate on important information within documents.

Drag & Drop Functionality

Users are able to move individual or multiple pages from a document into a new or existing document by simply dragging the thumbnail(s) into the desired tab. Allowing the user to move and rearrange pages within a single viewer across multiple tabs simplifies document manipulation and creation.

Enhanced Annotation Display

The viewer displays user information on each annotation, including the date and time stamp for when the annotation was made.

v4.5

Drag and Drop Page Manipulations

Users can easily reorder pages in a document simply by clicking on the page (or pages) in the thumbnail panel and dragging to the desired location.

Batch Redaction Tagging

Using the viewer's search and redact feature, users are able to tag an entire batch of redacted search results at the same time, rather than having to individually tag each redaction, expediting the workflow process.

Enhanced Cache Capabilities (Java only)

VirtualViewer's server caching has been redesigned to further boost performance, allowing users to greatly reduce repository processing times. The larger the document (100+ pages), the more noticeable the performance enhancement.

Upload Documents

Users have the ability to import local files directly from their computer into the viewer and decide whether to save directly into the repository/backend system or to keep them local.

v4.4

Search Annotation Text

The viewer provides users the ability to search for text though all text-based annotations in the current document using the Search tab in the Thumbnail panel, making collaboration on documents even easier.

Consolidate Annotation Layers

Users can consolidate all annotation layers of a document into a single layer so all annotations can be easily viewed.

Crop Page Selection

The viewer gives the user the ability to select a specific portion of a page using a rectangle tool to crop out the rest of the page. The cropped portion outside of the selected area is deleted from the page and the selected area can be saved out using "save as" or export.

Page Rotation Capabilities
Users are able to rotate specific pages as needed, making it easy to view documents and images as desired.

v4.3

Annotation and Redaction Tagging
Annotation and redaction tools allow multiple users to collaborate on a single annotation. Users are able to assign a tag (e.g. "Social Security Number") to each individual annotation or redaction to indicate to other users why the annotation or redaction was placed on the page.

Bookmarks
The bookmarks feature streamlines navigation within documents by providing users with the ability to create text bookmarks on pages via the thumbnail panel and also jump to a desired page via a bookmarks list.

Annotation Indicators and Navigation
Indicators ensure more efficient collaboration as users can now navigate through only the annotated pages of a document, skipping pages with no annotations or stamps.

Annotation Commenting
This workflow collaboration enhancement allows users to communicate about a specific part of the document by allowing comments to be added to existing annotations. Date, time of the comment, and the commenter's name are also listed.

v4.1

Redactions
Redactions streamline workflow while also ensuring sensitive data such as social security numbers and credit card information remains secure. The viewer provides users with multiple options for making redactions. Users can manually redact any region, highlight a specific selection of text, or search for a specific term. Once the sensitive information has been identified and marked for redaction, the user can then export a redacted version of the document, which is saved back to the document repository.

Document Notes
Users can add document notes to any document in order to maintain an active dialog and conversation within a specific document with other users. The notes are associated with the entire document (and not with specific pages) so collaborators can quickly review notes and action items.

User Preferences
A framework for viewer preferences allows users to customize VirtualViewer directly to their unique needs by concealing or displaying specific tools and functions. By hiding unused options, the user enjoys a cleaner interface with only the required functions taking up valuable screen real estate. The ability to determine default settings associated with annotations can expedite a workflow process and reduce processing errors.

v4.0

DWG Support
The addition of DWG and DXF to Snowbound's extensive file format library for VirtualViewer HTML5 .NET allows designers and architects to view CAD documents from any device with a web browser regardless of their location. CAD documents are typically used for engineering diagrams and blueprints.

SVG Support
The release also includes SVG support for the .NET viewer so users receive high resolution display at any zoom level when viewing extremely large documents. Snowbound developed its own SVG format conversion technology to improve viewing fidelity as well as improve performance by reducing memory requirements compared to traditional raster documents.

VirtualViewer® v4.9 New Features and Corrected Issues

Note that the Release Notes, separately packaged, have the most up to date descriptions of these features.

Documentation Corrections

- VirtualViewer for Java supports JRE 1.8. Previous documentation indicated that JRE 1.7 was supported.|
- For Windows products, .NET framework versions 4.5.2 and up are now supported.|

Add Public API for setting Username in User Preferences

API: virtualViewer.setUsername(string);

This was added so that the user can programatically add a username to their instance of VirtualViewer if they so desired. The user can still use the dialog box in the User Preferences.

Take advantage of new auto-growing version of IMG_save_bitmap

A better method for this function that takes advantage of RasterMaster's (Snowbound's Imaging Libraries that underlie VirtualViewer) new handling of buffer size and size increment values. It makes handling images of various sizes work better. It happens automatically.

Remove "Save as PDF" option in VV Print Menu

This option was removed because it has become a redundant feature that is now handled with the "Export" feature.

Footers for Page and Document thumbnails

Page and document thumbnails now display the file name in a footer.

Revalidate cache method called for every page

This is a short timespan cache to store answers from validateCache for each session/user. Every x minutes the cache will be deleted for each user (with storing and retrieval handled separately). This provides performance benefits to some users.

For whatever the specified window is (zero will check every time) we will cache the validation for that amount of time based on sessionId, documentId and HTTP action (GET or PUT). Once that time elapses, we will revalidate.

The time span value applies to both storage and retrieval.

VirtualViewer Java and .NET have slightly different parameters:

Java: The validation cache is defined in ehcache.xml with the document cache in a section for "vvValidationCache". By default, validations will expire after five minutes, although that is configurable in ehcache.xml.

.NET: There is a new initialization parameter "validationCacheExpirationMinutes" to control validation expiration. The default is five minutes.

Add ability to resize magnifier window

The ability to resize the Magnifier window vertically or horizontally using the mouse was added. To resize, grab the bottom left corner (a little black triangle) of the box.

Original magnifier size is defined in Config.js.

Document Notes Indicator

A red checkmark Document Notes indicator on the **Notes Tab** will toggle on if a document contains a document note, otherwise it will be toggled off.

Add Document Notes templates

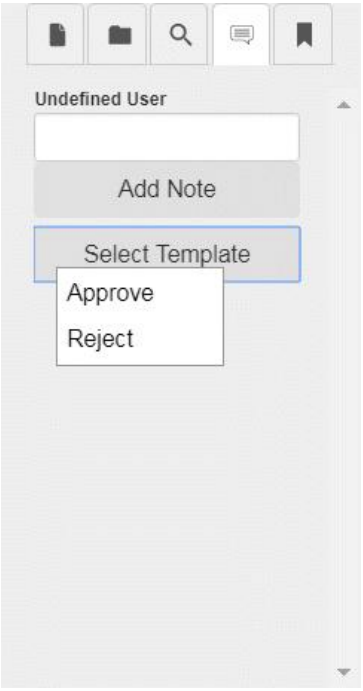
The ability to create Document Notes templates was added. The user can create a Document Notes template in two ways, either by adding the Document Note templates in **User Preference > Notes Templates** tab or by adding the template objects to the “noteTemplates” in **config.js**.

Follow the steps below to add a Document Notes template in User Preference:

1. Select the **User Preference** button**.** From User Preference dialog box, choose **Notes Templates** tab.
2. Select **Add** button to create new Document Notes template.
3. In **Template Name** field, enter the **template name**.
4. In **Template Text** filed, enter the **template text.**
5. Select **Save** button to save the template
6. To edit the Notes template, select the template then edit the **Template name** or **Template Text** field. Select **Save** button to save update template or **Cancel** button to exit.
7. To delete a template, select the template then select **Remove** button. Select **Save** button to save change or **Cancel** button to exit.

Document Notes Template workflow:

1. Select **Notes Tab**
2. Left-click **Select Template** button
3. Choose a template from the template drop down menu
4. Select **Add Note** button to add template to the document note.



Improve text copy/paste by new execCommand (copy) JS API

Clicking copy on the context menu now copies text to the clipboard without a modal popup. Ctrl+shift+c does the same (this is the hotkey defined in config to be copy). Old functionality is preserved just in case, as a fallback.Need more information

OCR Integration

This is the beta version of the OCR option in VirtualViewer. The final version is expected in the VirtualViewer v4.10 release. It is expected that a choice of OCR recognition engines will be provided in that release.

The OCR function allows searching text in an image document (TIFF or PNG initially) as well as selecting text in the VV client after the document has been OCRed. To OCR a document in the VV client, a user must search for text in a non-text document to get the OCR prompt. The OCRed result is cached; while that result is cached, the user can search for and select text without a further OCR prompt. Searching is performed using the Search tab in the thumbnail panel.

The original image will overlay the OCR'ed textual data to maintain the greatest similarity to the original document. The search text string will be highlighted. "Previous" and "Next" match buttons will work as normal. "Redact" and "Redact All Matches" work as normal. Applying redaction tags to results works as normal.

A wait icon will be displayed while the OCR process is running.

OCR will not be initiated if the input document is not PNG or TIFF raster. Saving to a PDF file is an option. Additional language support can be added by the customer.

The two new parameters in web.xml (web.config for .NET) are:

- ***enableOcr***: Enable OCR for searching and text extraction. Must have a valid OCR configuration and licensing to function correctly. Defaults to false.
- ***tesseractDataPath:*** Absolute or relative path to Tesseract OCR Engine's training data. If using packed WARs in Tomcat, this needs to be changed to an external unpacked folder. Defaults to "/tessdata".

Add Watermark Support in VirtualViewer

Overview:

VirtualViewer now offers watermarks for customers who need to mark page backgrounds with specific notifications such as "Private", "Confidential", "Do not distribute" and so on. Watermarks can be created that are transparent or solid, of varying fonts and sizes and positions. They can also be restricted to admins versus all users.

They can also have dynamic tags for user name, page numbers, print time, and document name.

What does the User Interface look like?
There are a few UI changes. A new watermarks dialog lays out all the watermarks options for creation, deletion and editing. In the "document handling" dialogs (printing, exporting, etc) a new checkbox has been added, so the user may decide whether to burn their watermarks when exporting.

If a watermark is marked as admin-created, then the "burn watermarks" option will be checked and disabled, so the admin watermarks burn by default. Similarly, a non-admin may not edit or delete admin-created watermarks.

Those dialogs are the only way to interact with watermarks. You can't select them on the document, move them around, etc--they're not annotations, they're marked into the document once created.

Other features
Users may add dynamic data into their watermark text. This is easily done in the watermarks dialog by clicking on a tag button above the text box in the watermark dialog. If you inspect the raw text of the JSON, a tag will appear enclosed in two @ signs, which may be escaped by adding a /. When displayed, the tag will be replaced by data.

For instance, the user wants a page number to print on each page. They click the tag button in the dialog. In JSON, now the watermark text would say, "Page @@pagenumber@@". When displayed on the document, the watermark on page one will read "Page 1", the watermark on page fifty will read "Page 50" and so on. If the user types "If I wanted a page number I would use /@/@pageNumber/@/@", the watermark will now display "If I wanted a page number I would use @@pagenumber@@". The tag is escaped, and so is not replaced by a dynamic number.

Available tags are:

- Username: the user's username as stored in user preferences.
- total pages: the number of pages in the document.
- current page number: the number of the current page.
- print time: The date & time when the document was exported or printed. When displaying in the viewer, this is just an example date and time, from when the document was opened.
- document name: The display name of the current document.

Watermark JSON Files
Watermarks for a document are stored in a json file. Like annotations, the file will be documentkey + suffix. For instance, 6-Pages-1.tif.watermarks.json. The .watermarks.json file is a list of json objects, so it has the format:

[{ myJsonData }, { myOtherJsonData }].

Each individual watermark is a json object. Each will have the following properties, formatted as seen in the attached example:

- *transparency*: A boolean. If true, the watermark will be transparent; if false, it will be a solid color.
- *adminCreated*: A boolean. If false, any user can manage any aspect of the watermark. If true, admin restrictions will apply (as described below).
- *text*: A string. This is the text that will appear on the watermark. Multiline watermarks are supported. This is done under the hood in the watermarks dialog, but if a user is manually entering json, they should enter a newline character ("\n") where a line break should be.
- *allPages:* A boolean. If this is set, the watermark will appear on every page of a document.
- *pages:* An array of page indices, zero-indexed. For instance, to place a watermark on only page one, this property would contain [0]. This is a key difference between watermarks and annotations. Watermarks are intended to repeat across pages, so an identical watermark will have multiple pages it applies to.
- *widthAtTenPx:* An integer. This is a read-only value used by VirtualViewer to calculate the dimensions of the watermark, representing how wide the watermark is when the font is 10 pixels high.
- *stretch*: A double. This defines how far across the page the watermark will stretch. Set to 1.0, the watermark will go across 100% of the page (minus some margin space). Set to 0.5, 50% of the page. The UI allows only a small set of percentages. Diagonal watermarks will always stretch 100% across the diagonal.
- *format*: A json sub-object that has font and color information, as follows.
  -- font: A font name, for instance "Arial".
  -- color: We currently support only one color, so "000000" would be stored here.
- *position*: This is another sub-json object, that defines where the watermark will be placed on the page. There are two defining properties in here: the vertical placement of the watermark (top of the page, middle of the page, or the bottom of the page) and the direction of the text. While these options may open up further, the direction options are currently left-to-right text or diagonal text. The two options combine so that, for instance, top vertical placement & diagonal direction produce a watermark stretching from the top-left to bottom-right corner--while bottom vertical placement & diagonal direction will go from bottom-left to top-right.
  -- vertical: Use 0 for top, 1 for center, and 2 for bottom.
  -- direction: Use 0 for left-to-right text, and 2 for diagonal text.

Watermark.json file sample

[{"widthAtTenPx":19,"transparency":true,"adminCreated":false,"text":"bugs","allPages":true,"pages":[],"stretchPercent":0.5,"format":{"font":"Times New Roman","color":"000000"},"position":{"vertical":0,"direction":0}},{"widthAtTenPx":86,"transparency":true,"adminCreated":false,"text":"second%20watermark","allPages":false,"pages":[0],"stretchPercent":1,"format":{"font":"Times New Roman","color":"000000"},"position":{"vertical":2,"direction":2}},{"widthAtTenPx":62,"transparency":false,"adminCreated":false,"text":"sdadafsadfgsafd","allPages":false,"pages":[0],"stretchPercent":1,"format":{"font":"Times New Roman","color":"000000"},"position":{"vertical":2,"direction":0}}]

Last minute changes on Watermark feature (will be corrected in a point release soon).

Unicode will not be supported in this release.

Getting Started

Snowbound Software's VirtualViewer HTML5 for .NET works with the latest .NET technology to create a true zero footprint viewing solution. This document will aid you with setting up and working with the package included in your zip file, **virtualviewer.zip**. This zip file installs all of VirtualViewer HTML5 for .NET components on a single system. For information on configuring VirtualViewer HTML5 for .NET, please see Using VirtualViewer HTML5.

System Requirements

Supported Operating Systems (Server - 64 bit)

Internet Information Services (IIS) 7.5 or higher.

Windows Servers 2016, 2012 R2, and 2008 R2 (Server 2019 coming soon)

Recommended only for Proof of Concepts:

- Windows 10
- Windows 8
- Windows 7

Supported Browsers

Google Chrome latest through 42

Microsoft Edge

Firefox latest through 35

Safari latest through 8

Internet Explorer 11 (special circumstances)

Note if using Internet Explorer 11:
VirtualViewer will look, perform, and behave better if it is running outside of compatibility mode in Internet Explorer 11. For best performance, please configure Internet Explorer to use normal mode when using VirtualViewer. Quirks mode in Internet Explorer is not supported.

*Some functionality is limited.

Important Note:

Exceptions to Supported File Formats and Platforms

Exceptions to Supported File Formats and Platforms

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

Validation Minimum Requirements

The following are the validation minimum requirements:

	Minimum Requirements
Processor	64bit
Speed	2.4 GHz dual core
Ram	16GB
Available Memory	6GB
SSD or HD Space	250GB

Performance Testing Requirements

For performance testing, the following minimum and recommended server requirements are suggested.

Note:
During performance testing, we recommend launching the VirtualViewer client on a separate machine as the Application Server.

Minimum Server Requirements

The following are the minimum server requirements:

	Minimum Requirements
Processor	64bit
Speed	3.2 GHz dual core
Ram	32GB (see below)
SSD or HD Space	250GB

Recommended Server Requirements

The following are the recommended server requirements:

	Recommended Requirements
Processor	64bit
	3.4 GHz quad core (with hyper-threading)
Ram	64GB (see below)

	Recommended Requirements
SSD or HD Space	250GB

Determining Memory Requirements

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

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

- You go from black and white, to grayscale, to color documents (bits per pixel increases).
- You go from compressed to uncompressed document formats (lossy compression to raw image data).
- You go from low resolution to high resolution documents (dots per inch / quality increases).

You go from small index card size images to large blueprint size images (number of pixels increases).

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

Decrease the number of documents you have open at any one time.

Decrease the quality of the images requested by decreasing bits per pixel, the resolution, or the size.

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

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

(height_in_pixels * width_in_pixels * (bits_per_pixel/ 8)) = image_size_in_bytes

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

Table 1.1: Memory Requirements Based on Image Size

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

Determining Memory Needed for the Number of Users and Pages Viewed in VirtualViewer® HTML5

To calculate the amount of memory needed based on the number of users and potential pages viewed at any given time, use the example below:

The number of concurrent users * size per page in MB * 5 pages in view

For example, plug in the number of pages (in this case, 5) and the number of users (in this case, 1000):

black and white page (100 dpi) .1mb per page x 5 pages= .5 mb x 1000 users = 500 mb =~ 0.5 GB

black and white page (300 dpi) 1mb per page x 5 pages= 5 mb x 1000 users = 5000 mb =~ 5GB

color pages (300 dpi) 25mb per page x 5 pages = 125 mb x 1000 users = 125000 mb = ~122 GB

Licensing

VirtualViewer HTML5 for .NET is delivered as a .zip file (virtualviewer.zip) including the installer module virtualviewer.msi.

After you unzip virtualviewer.zip, you will find the virtualviewer.msi installer files for you to install.

Your options are enabled through a contains a license.xml file

The most current set of documentation is included with the installation package to assist you in installing and administrating this product. The documentation is described below and can be found in the Documentation directory within the .zip file or online at www.virtualviewer.com.

> **VirtualViewerHTML5NetAdminGuide.pdf**: This guide describes how to use and configure VirtualViewer HTML5 for .NET.
>
> **VirtualViewer x.yy ReleaseNotes.pdf**: The release notes describe the latest additions and improvements to VirtualViewer HTML5 for the x.yy version.

Location of the Log Files

For VirtualViewer HTML5 for .NET, you can find the logs files at the following locations if you are using the non-integrated production installation:

`virtualviewernetcontentserver.log`at Install Directory \VirtualViewer

For VirtualViewer HTML5 for .NET, you can find the log files at the following location if you are using the integrated installation:

`virtualviewernetajaxserver.log` at InstallDirectory\Virtualviewer

If log files do not appear, be sure to add write permissions to the respective directories.

What to Expect in an Evaluation Version of VirtualViewer® HTML5

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

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

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

What to Expect in a Production Version of VirtualViewer® HTML5

When you purchase VirtualViewer HTML5 for .NET, you will receive a set of fully licensed binary files. The files will include snbd*.dll and .lib + .dlls. The .dlls must be placed in the bin directory that contains SbdNetAnn.dll.

Note:
Please back up any files that were modified during the evaluation, then place those files back in the VirtualViewer directories after installing the production version.

If the viewer does not display DOC, PDF and DWG files as expected, please **restart IIS** and your **Windows system**. When the system Path environment variable has been modified to include the Snowbound installation directory

specifying the location of the DOC, PDF and DWG plug-ins, the process that uses the Path variable needs to restart to pick up the new Path value.

Installing the Production Version of VirtualViewer® HTML5

When you receive your production version, you can extract the files from the production version package and use those to replace the same files in the evaluation version that you have installed. It is also possible to substitute your new Production license file for the previous license file if the other files are the same, or you want to initially start with older files.

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 have questions or problems, feel free to contact us at http://support.snowbound.com.

Installation Checklist

To properly run VirtualViewer HTML5 for .NET, it is critical that all minimum environment requirements are met and installation steps are followed exactly.

Before running the installation package, make sure that the operating system, IIS version, .NET framework, and browser(s) being used are all supported by this product. Although this product may work to a certain capacity with different environments from the ones listed, only the environments listed have been fully tested and can be fully supported.

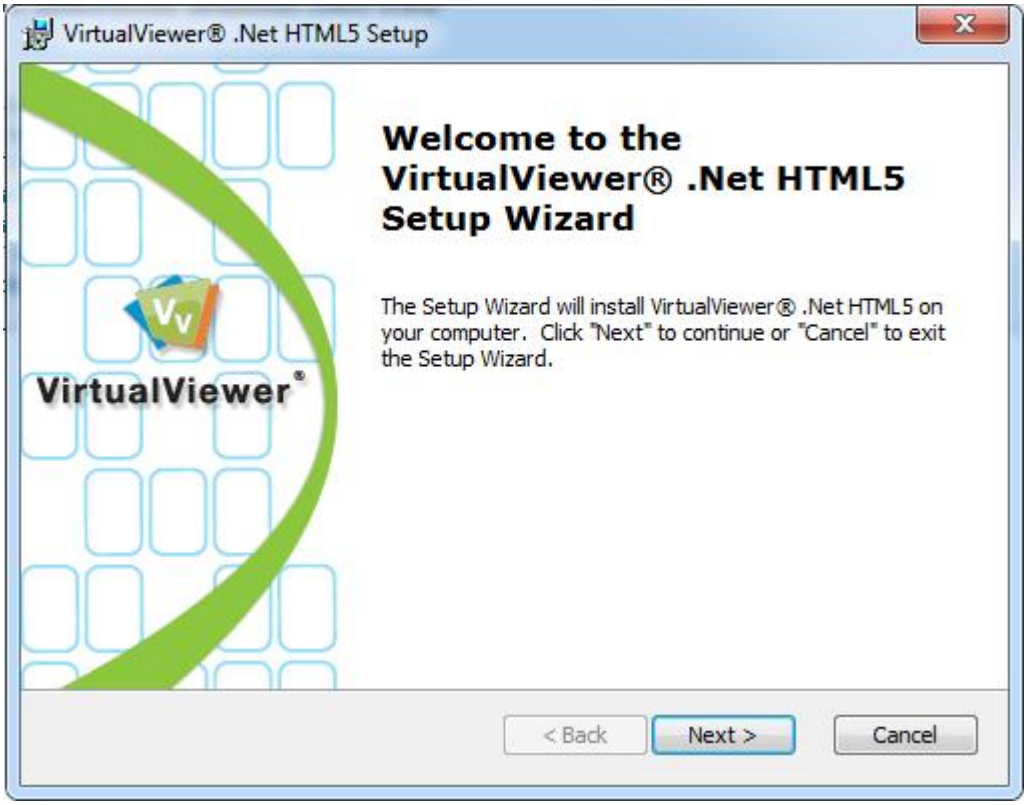
The typical installation will install to your IIS directory and will require no further configuration to run in any supported environment.

Both the content server and .NET Server write logs to their respective directories. It is critical that these directories be given proper permissions and security settings. For all supported environments, both the local **IUSR** and **IWAM** accounts will need to be given full write permissions to these directories. Depending on your general security settings on your server, you may also need to give write permissions to other local and network accounts.

Installing

To install VirtualViewer HTML5 for .NET, follow the steps below:

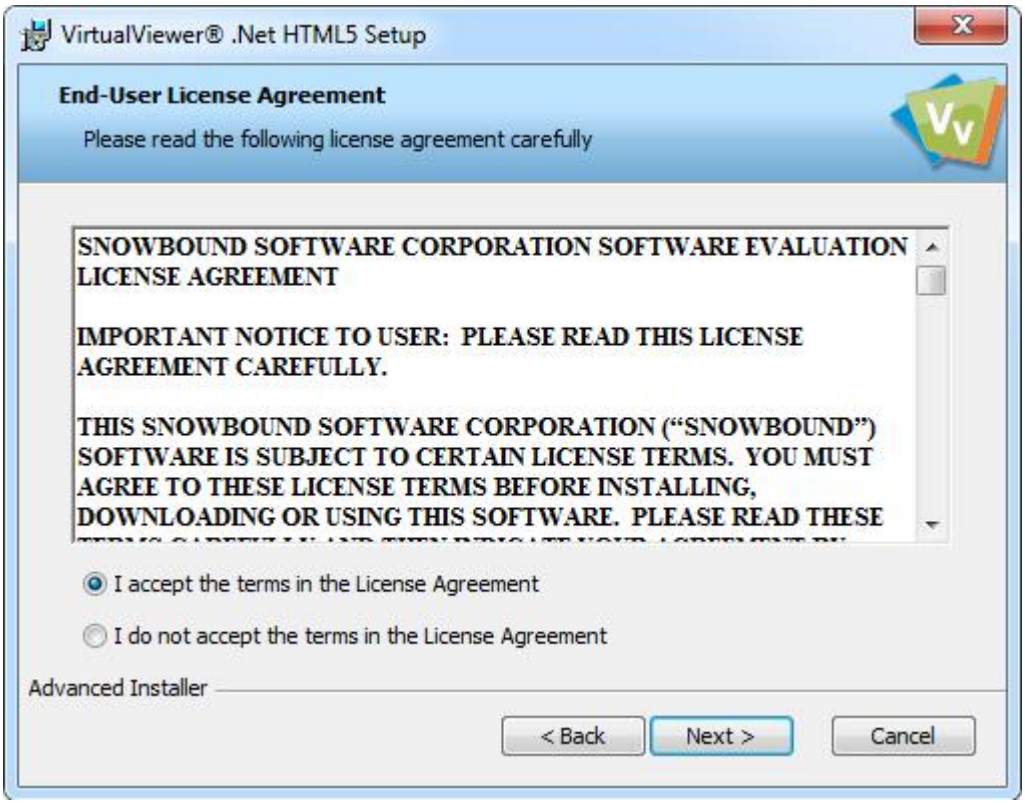
1. Double-click on the downloaded .zip file. In this example, double-click on **VirtualViewerNetHTML5.zip**.



VirtualViewer HTML5 for .NET Setup - Welcome Dialog

2. Select the appropriate icons to customize the way the features are installed.

Note:
Select the **Disk Usage** button, to see how much disk space is available. Any highlighted volumes shown do not have enough space for the installation. You can either remove some files from the highlighted volumes or install less features. Select the **Reset** button, to cancel the selections that you have made and reset the options to their original settings. Select the **Next** button, to proceed with the installation.

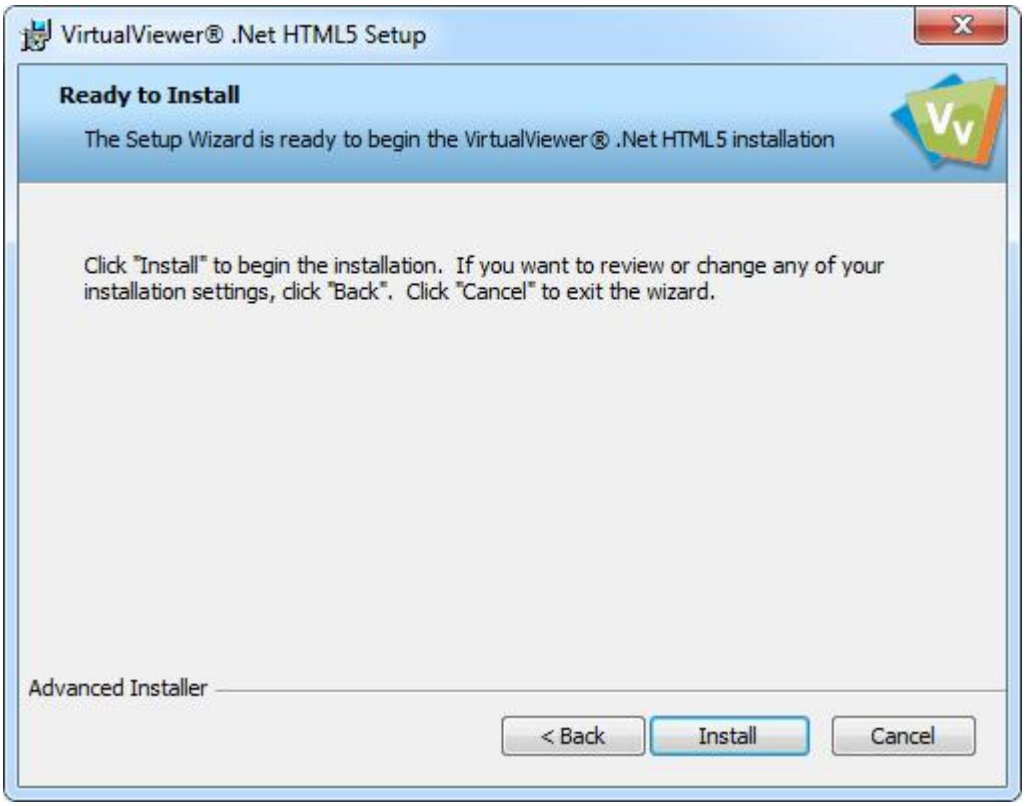


VirtualViewer HTML5 for .NET Setup - License Agreement Dialog

3. Read the license agreement.

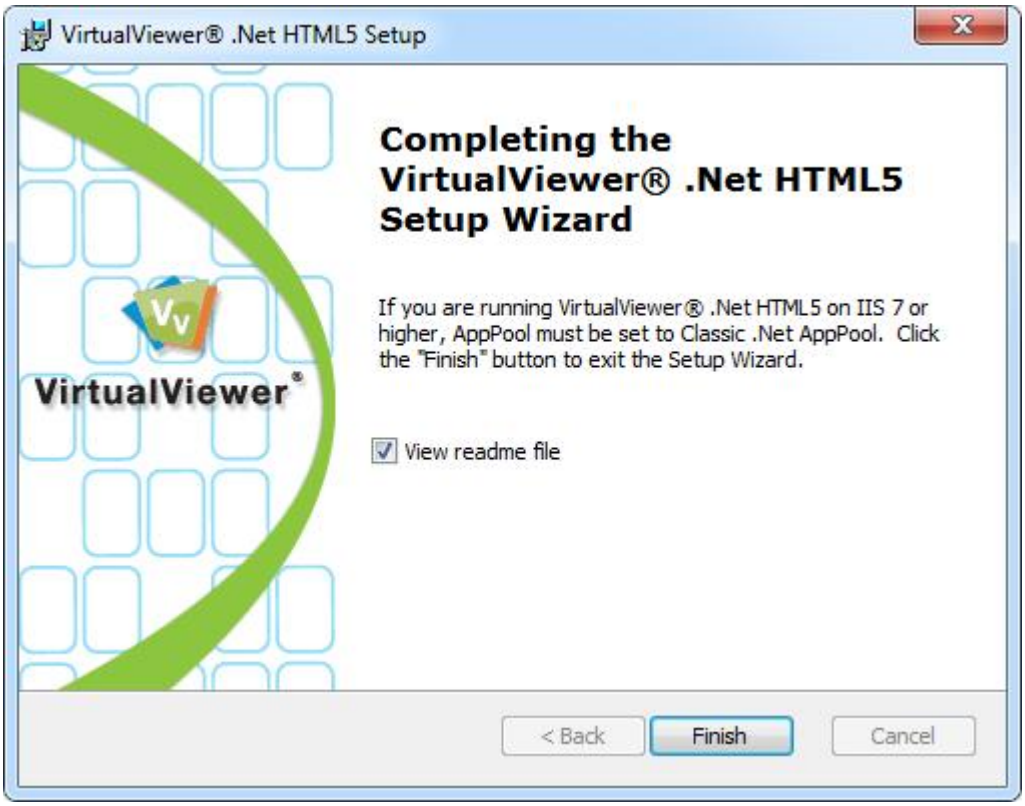
Note:
If you **agree** with the license agreement, select "I accept the terms in the license agreement" and click **Next** to proceed with the installation.

Note:
If you **do not agree** with the license agreement, you cannot proceed with the installation.



VirtualViewer HTML5 for .NET Setup - Ready to Install Dialog

4. Click **Install** to begin the installation.



VirtualViewer HTML5 for .NET Setup - Completing the VirtualViewer HTML5 for .NET Setup Wizard Dialog

5. Click **Finish** to complete the installation.
6. Click the **View readme file**, to view the read me file.
7. The “Thank you for installing VirtualViewer HTML5 for .NET“dialog displays. You have successfully installed VirtualViewer HTML5 for .NET. Your files are installed to the default location of your IIS directory. This directory is generally found at `C:\Inetpub\VirtualViewer`.
8. Please see the documentation. This is generally found with the installer or on the website.

Note:
If the viewer does not display DOC and PDF files as expected, please **restart IIS** and your **Windows system**. When the system Path environment variable has been modified to include the

Snowbound installation directory specifying the location of the DOC and PDF plug-ins, the process that uses the Path variable needs to restart to pick up the new Path value.

If you ran the default installation, all of your files will have been installed to the default IIS directory. If your server is configured to run IIS, you should now be able to run VirtualViewer. You can use the following URL on your server to test the installation:

```
http://-
localhost:8047/VirtualViewerNetHTML5/index.html?documentId=snowLogo.jpg
```

The default port for VirtualViewer .NET is **8047**. To configure a different port, set the **codebase** and **serverURL** parameters in the web.config file.

For example, change the port 8047 in the example below to your port number.

```
<add key="codebase" value="http://localhost:8047/VirtualViewer"
/>
```

```
<add key="serverURL" value- e="http://localhost:8047/VirtualViewer" />
```

If you ran the production installation and your server is configured to run IIS, you should now be able to run VirtualViewer. You can use the following URL on your server to test the installation:

```
http://localhost:8047/VirtualViewer/
index.html?documentId=VirtualViewerHTML5NetAdminGuide.tif
```

If the viewer does not load the test document on your first attempt, please wait a minute and try again. IIS can often take a few minutes to fully adapt to any modifications and/or configurations.

For information on how to install IIS7, please see the following link:

http://technet.microsoft.com/en-us/library/cc731911(WS.10).aspx

If you are running VirtualViewer on Windows 10, you need to uncomment the following lines in web.xml and create both folders. This is only required if you are manually installing it. The installer will take care of this for you.

```
<!-- Uncomment both for Windows 10 and make sure both directories exist -->
<add key="libreOfficeProfilePath"
value="C:\inetpub\snowbound\virtualviewer\LibreOfficeProfile" />
<add key="libreOfficeTempPath"
value="C:\inetpub\snowbound\virtualviewer\LibreOfficeTemp" />
```

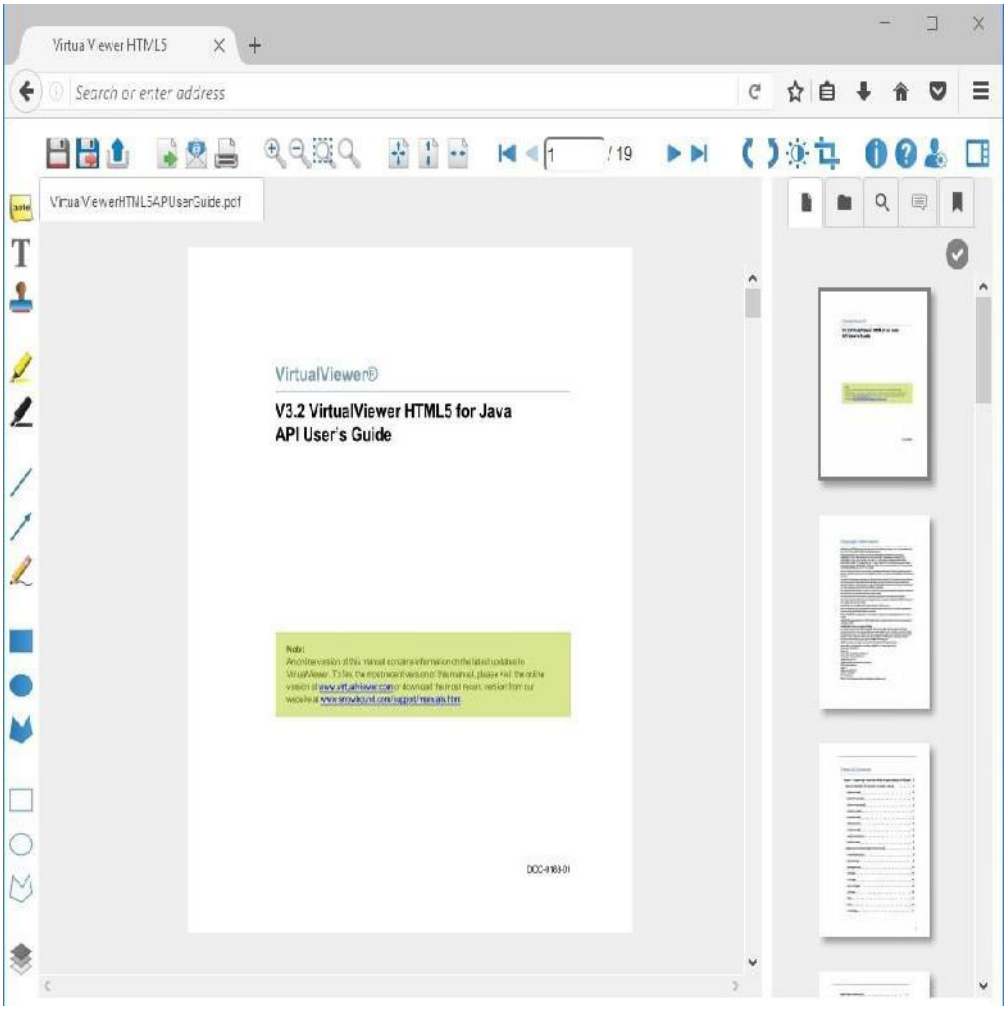
Verifying

Running VirtualViewer® HTML5 in a Browser

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

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

The following example shows VirtualViewer HTML5 for .NET loaded in a browser:



Verifying that Your Documents Work in VirtualViewer HTML5 for .NET

Snowbound Software provides some sample documents in the VirtualViewer HTML5 for .NET installation to get you started. The sample files are located in the **Sample-Documents** subdirectory. The web.config file delivered with VirtualViewer HTML5 for .NET is located in `VirtualViewerNetHTML5` specifies the Sample-Documents subdirectory as the default location of the sample files in the `filePath` parameter.

To view the sample documents, enter the URL shown in the **Displaying the Sample Documents** example below.

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 Snowbound test document.tif, add that name to after `documentId` as shown in the following example:

Example 1.1: Displaying the Sample Documents

```
http://servername:port/Vir-
tualView-
erNetHTML5/index.html?documentId=Virtualviewer5NetAdminGuide.tif
```

For example:

```
http://localhost:8047/VirtualViewer/
index.html?documentId=Virtualviewer5NetAdminGuide.tif
```

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

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

Example 1.2: Specifying the Document to Display

```
http://localhost:8047/VirtualViewer/index.html?
documentId=Virtualviewer5NetAdminGuide.tif
```

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

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

For more information, please see Connecting Your Document Store.

Using VirtualViewer HTML5

This section describes the available functionality and features in VirtualViewer® HTML5.

The first three sections describe the functional areas of VirtualViewer HTML5 for .NET. The Image Controls Toolbar runs along the top of the screen. The Annotation Toolbar that runs along the left side of the screen. The Pages and Documents 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.

The Image Controls Toolbar

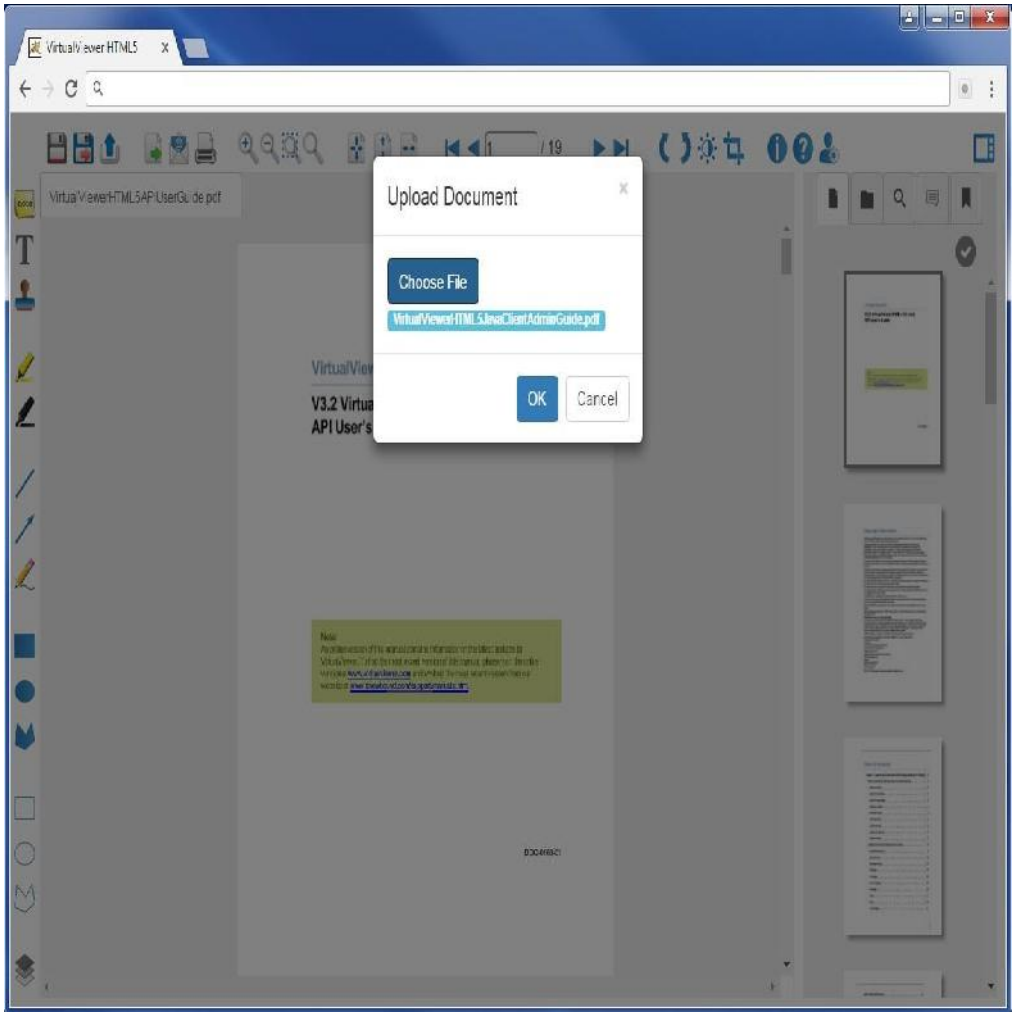
The section describes the Image Controls Toolbar that runs along the top of the VirtualViewer HTML5 for .NET screen.

Load Local Files with Upload Document

Load Local Files with Upload Document

Follow the steps below to use the load local files feature:

1. Select the **Upload Document**  button.
2. In the Upload Document dialog, select the **Choose File** button.
3. Select the file that you would like to open. The file name displays in the Upload Document dialog.
4. Select the **OK** button.
5. A copy of the file is uploaded to the server and is automatically placed in your Sample Documents folder to use in VirtualViewer.



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

Exporting a Document with Annotations

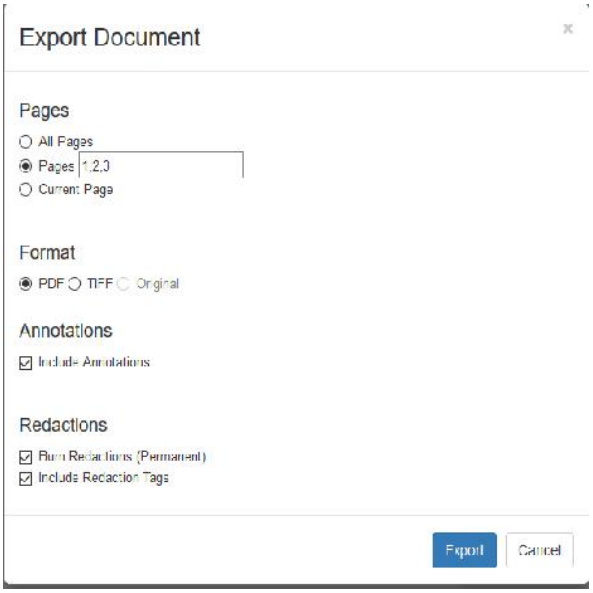
The Export dialog box contains the Include Annotations checkbox to select the option to export a document with annotations. Annotations will only be included when the **Include Annotations** checkbox is selected. The default is set to include annotations when exporting. When exporting with

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.

On the Include Annotations dialog box, the Text and Non-Text options are hidden from the Export Document, Email Document, and SaveAs Document sub-option. We still support these options. To re-enable the Text and Non-Text option, change their respective entries from “display: none” to “display :! Important” in dialog.css.

```
div#vvExportOptionsAnnotationsTypeCheckboxes {
display: block !important;

}

div#vvEmailOptionsAnnotationsTypeCheckboxes {
display: block !important;

}

div#vvSaveAsOptionsAnnotationsTypeCheckboxes {
display: block !important;

}
```

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.



Emailing a Document

To email a document, select the **Email** button. 

The Email Document dialog box appears. Select the options that you want for your email.

In the **From:** field, enter the email address of the sender.

In the **To**: field, enter the email address where you are sending the document.

In the **Subject**: field, enter the subject for your email.

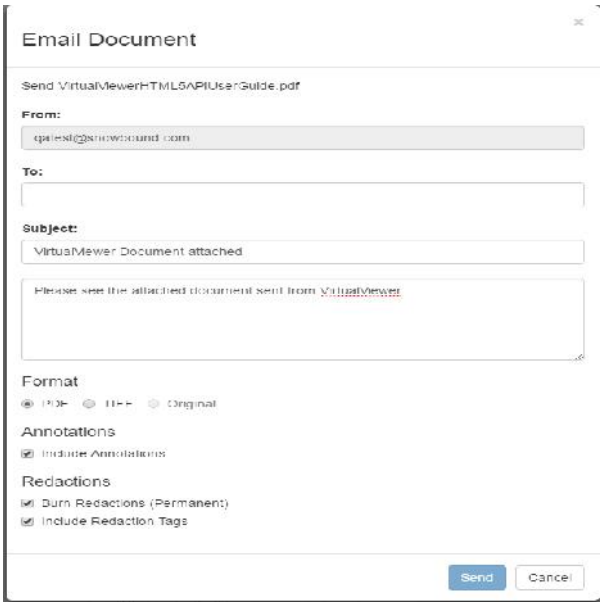
In the **body** field, enter the text of the email.

In the **Format** section, select PDF, TIFF or Original for the file format.

In the **Annotations** section, select any of the following check boxes:

Include Annotations - Check to include annotations.

Select **Send** to send the email.



Use the following web.xml servlet tags to set default values for sending emails:

Example 1.1: Setting the default email address

```
<init-param>
<param-name>emailFromAddress</param-name>
<param-value>address@company.net</param-value>
</init-param>
```

Example 1.2: Setting the default SMTP Server for Emails Sent via Email Document

```
<init-param>
<param-name>emailServer</param-name>
<param-value>servername</param-value>
</init-param>
```

Printing

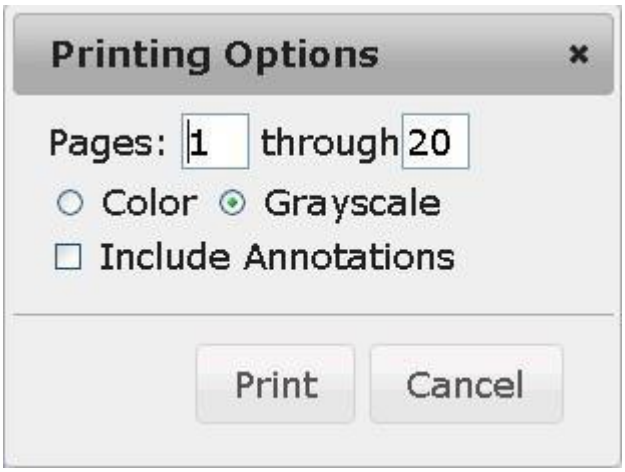
To print, select the **Print** button.

The Print dialog box appears. Select the options that you want for your print job. The “Save as PDF” option was removed because it has become a redundant feature that is now handled with the “Export” feature.

Printing 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 Displaying the Include Annotations Checkbox in Chapter 3, Customizing the Configuration.



Printing Text and Non-text Annotations Separately

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

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

Zooming

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.

Magnifier

To magnify, select one of the **Magifier** buttons:

Magnifier  .

When the Magnifier is launched, it appears on the screen based on the default coordinates defined in config.js.

Once the Magnifier is displayed, it can be selected and moved just like any annotation.

The Magnifier size does not scale with changes to the zoom level of the page and maintains its dimensions as the page zooms but the zoom will scale as a factor of the magnifier zoom level and the page zoom level.

The ability to resize the Magnifier window vertically or horizontally using the mouse was added. To resize, grab the bottom left corner (a little black triangle) of the box.

The magnifier will not magnify annotations.

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  .

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  .

Continuous Scrolling

The thumbnail panel scrolls as you scroll the document in the image panel. The page in the image panel that has greater than 50% of the available screen is reflected as the active thumbnail.

As you scroll through the document pages, the viewer automatically highlights the border of the thumbnails after the page has changed. The page number changes to reflect the page selected in the thumbnail.

Any page level calls are applied to the active page. For example, if you select to rotate, only the active page is rotated.

Any zoom level functions are applied to the entire document. For example, if you select, fit-to-page, every page in the document displays as fit-to-page.

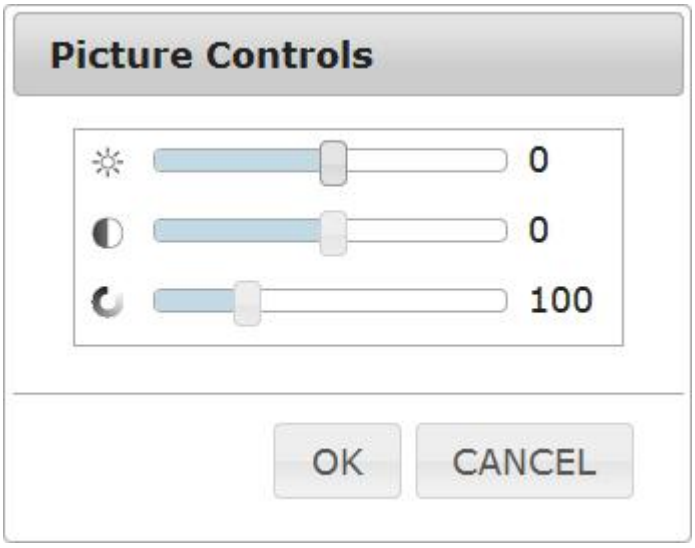
Continuous scroll facilitates searching. The found words can be highlighted in all the pages of the document.

Picture Controls

To adjust image properties (picture controls), select the **Picture Controls** but-

t

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



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

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

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

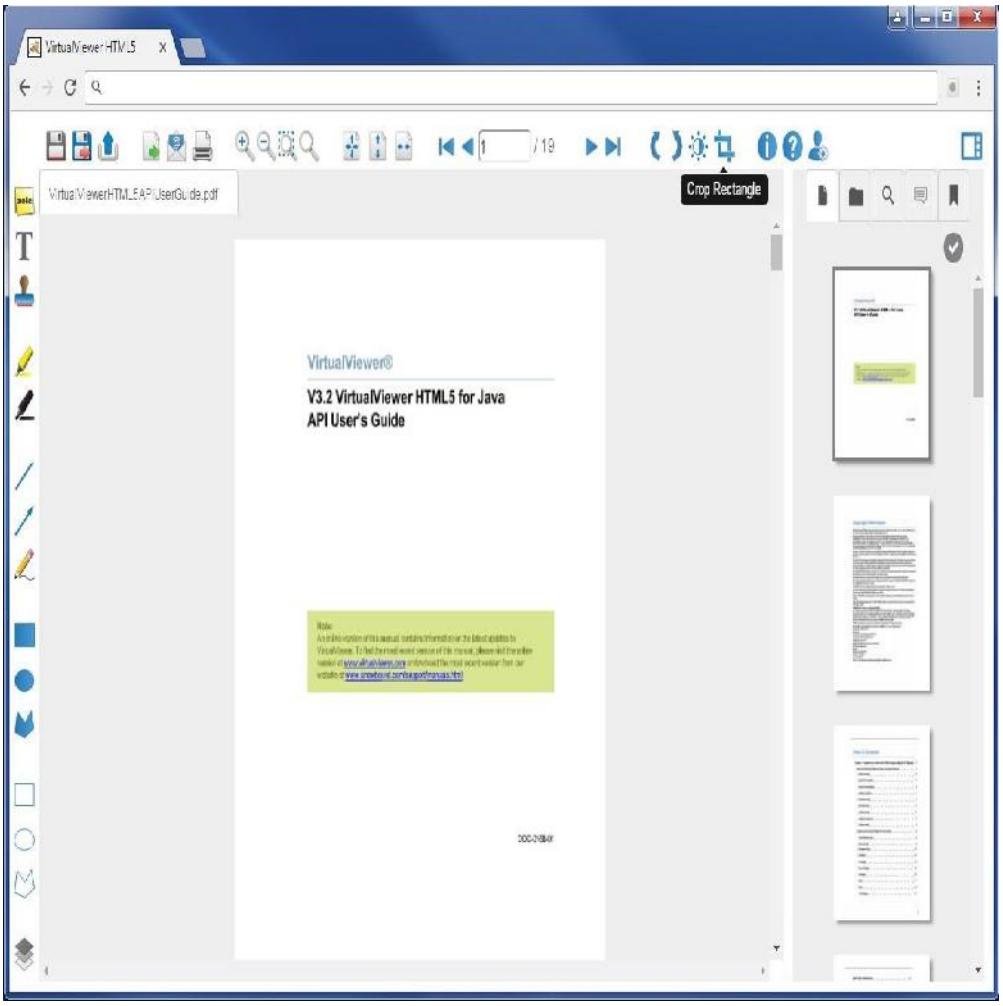
Changes made to the Picture Controls properties will be seen in the viewer, in near real time, as the adjustments are made.

Crop Page Selection

Crop Page Selection

You can draw a crop rectangle on a page and crop out to remove the rest of the page. The remaining area is deleted from the page and can be saved out using save as or export.

To crop a page, select the **Crop Page Selection** button. With the Crop Page Selection button, draw a rectangle to select the area that you want to crop from the page. Select **OK** to confirm the area to crop. The area outside the selection is deleted from the page. Select **Cancel** to cancel the selection. Select **Save** to save the cropped area.



Crop Page will not retain any annotations or pre-burned redactions.

When entering crop preview mode, cropping will remove all annotations from your page. In order to save the crop, either use export or save as to send it to a new document, or save to overwrite the file.

If you crop within a cropped page, the original crop will be backed out. You cannot crop a cropped area. You have to save the original cropped page to crop again.

Annotations are not supported on cropped pages. If you try to annotate on a crop preview page, you will see an error message and not be able to annotate the page. You can annotate after saving the cropped page.

In config.js, set the `enableCrop` parameter to true to enable consolidate annotation layers. Set the parameter to false to disable consolidate annotation layers. This parameter is set to false by default.

User Preference

The User Preferences feature allows you to configure the icons, annotation properties, text stamps, and the default fit-to display in the viewer.

Select the User Preferences icon  to open the User Preferences window with the following four panels to set the user preferences:

Panel 1: **Toolbar Configurations** sets the ability to show or hide each button on the Image Controls and Annotation toolbar.

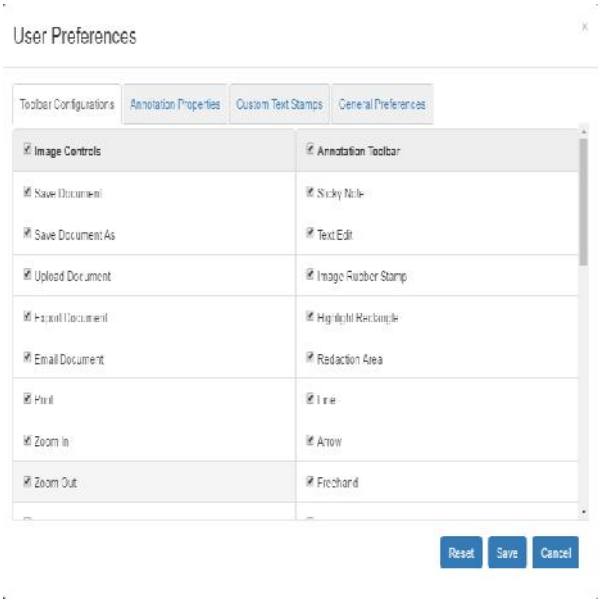
Check or uncheck the check box for the toolbar button that you want to turn on or off. Check the box to show the button on the toolbar. Uncheck to hide that button from the toolbar.

Check or uncheck the top check box to turn on or off toolbar icons.

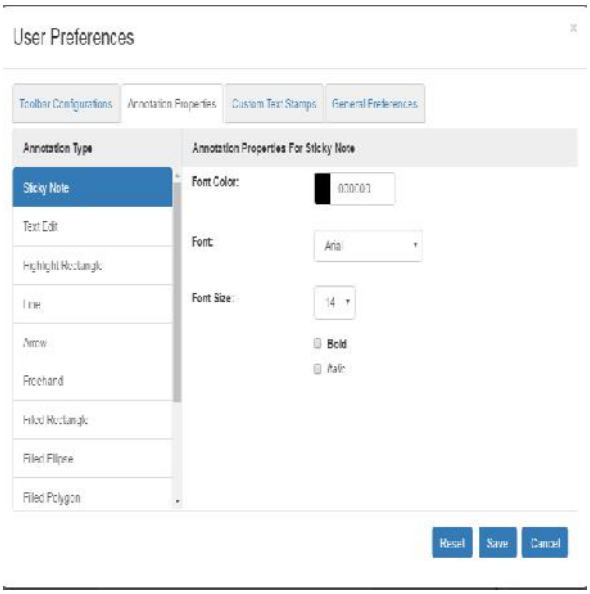
The icons will shift on the toolbar to fill in the space left by icons that are turned off.

The changes that you make in User Preferences are saved to your local storage on the browser that you used when making the changes. Your User Preferences settings will only be visible on the browser on the computer where you saved the settings.

Note:
Select the **ctrl + '** shortcut key to launch the user preferences dialog box. This is useful if you have turned off the User Preferences icon in the toolbar and want to open the User Preferences window.



Panel 2: **Annotation Properties** sets the default values for the annotation types. This includes the font type and size, font color, and line size.



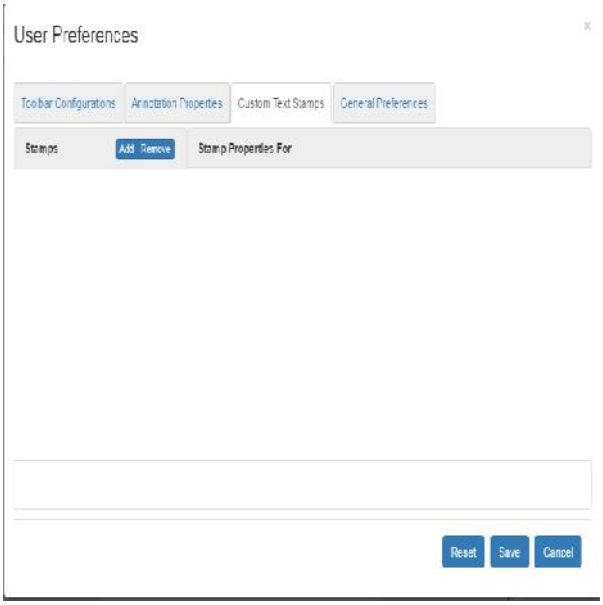
Panel 3: **Custom Text Stamps** defines custom text stamps. Select the [+] button to add a custom text stamp. Enter the display name and stamp text. Select the appropriate font color, font type, font size, bold or Italic text. Select the [-] button to remove a custom text stamp.

The panel shows a real time preview of the custom text stamp.

The **display name** shows the text that displays for the custom text stamp in the toolbar.

The **stamp text** displays the text that is displayed in the custom text stamp annotation.

Custom text stamp is disabled by default. To enable custom text stamp, set `enableTextRubberStamp`**config.js** parameter to true.



Panel 4: **General Preferences** sets the default fit-to preferences.

Select **Fit to Window**, **Fit to Height**, or **Fit to Width**.

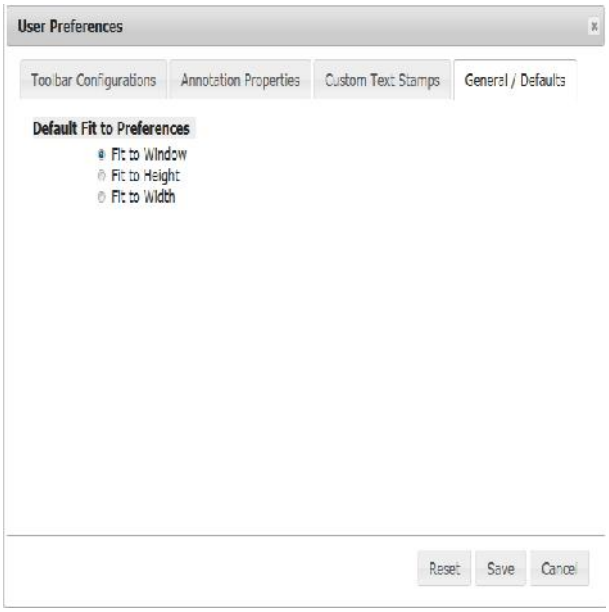
Select **Zoom Percent**. From the drop down select the zoom level from the following:

2, 3, 4, 6, 8, 10, 15, 20, 30, 40, 50, 75, 100, 150, 200, 300, 400, 600, 800, 1000.

In **config.js**, set the `zoomLevels` parameter to the desired zoom levels. The default values are 2, 3, 4, 6, 8, 10, 15, 20, 30, 40, 50, 75, 100, 150, 200, 300, 400, 600, 800, 1000.

Once the level is chosen and saved, the document will automatically switch to the selected zoom level. For example, if you choose 75, the document will zoom to 75%.

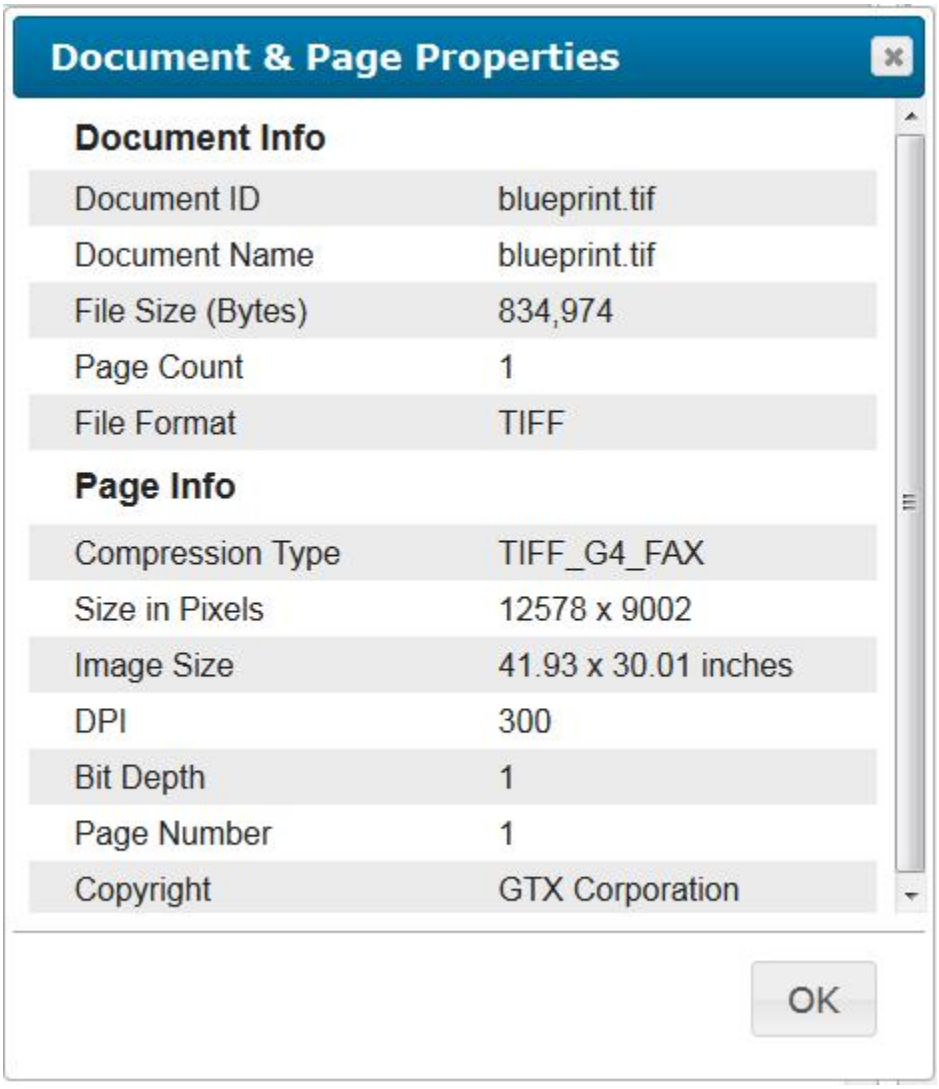
In the **Display Name** field, enter the user name to set a user name for Document Notes and Annotation Commenting.



Select the **Reset** button to clear local storage and remove the browser specific, user defined stamps. Select the **Save** button to save your user preferences in the browser cache. Select the **Cancel** button to cancel the window.

Display Document and Page Properties

Select the Image Info icon to open a window with the following document properties information:



Use the following API methods to manage the display of document properties:

Example 1.3: Showing the Information for a document

```
VirtualViewer.prototype.showImageInfo = function()
```

Example 1.4: Hiding the Information for a document

```
VirtualViewer.prototype.hideImageInfo = function()
```

Example 1.5: Toggling the Information for a document

```
VirtualViewer.prototype.toggleImageInfo = function()
```

The keyboard shortcut to toggle image information is: **CTRL+SHIFT+u**.

The Annotation Toolbar

The section describes the Annotation Toolbar that runs along the left side of the VirtualViewer HTML5 for .NET screen.

Creating Annotations

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

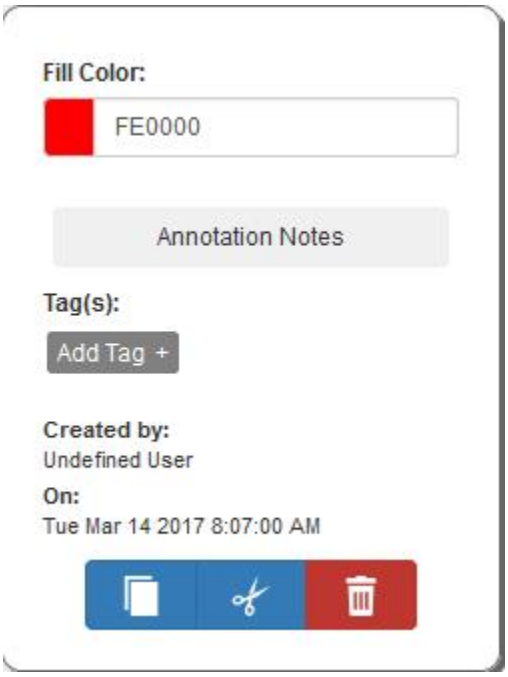
Note:
Annotations ARE supported on the iPhone and iPad platforms.

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

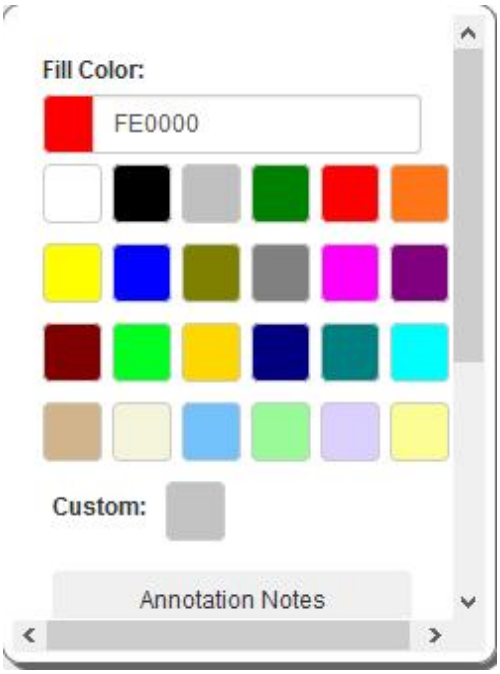
- Select a color to fill in the annotation.
- Select a line color.
- Adjust the line size for a line annotation.
- Edit the text for a text annotation.

Editing a Filled Annotation

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



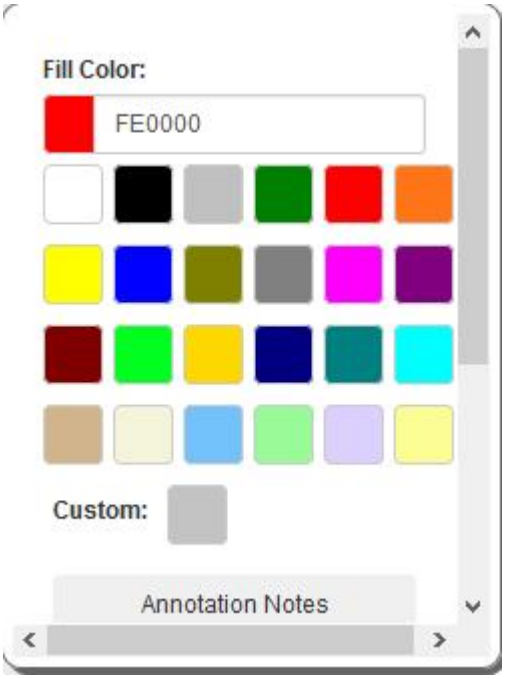
To display more fill colors, select the **More Colors...** link. The Fill Color box expands to display more colors to select from:



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

Editing a Line Annotation

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

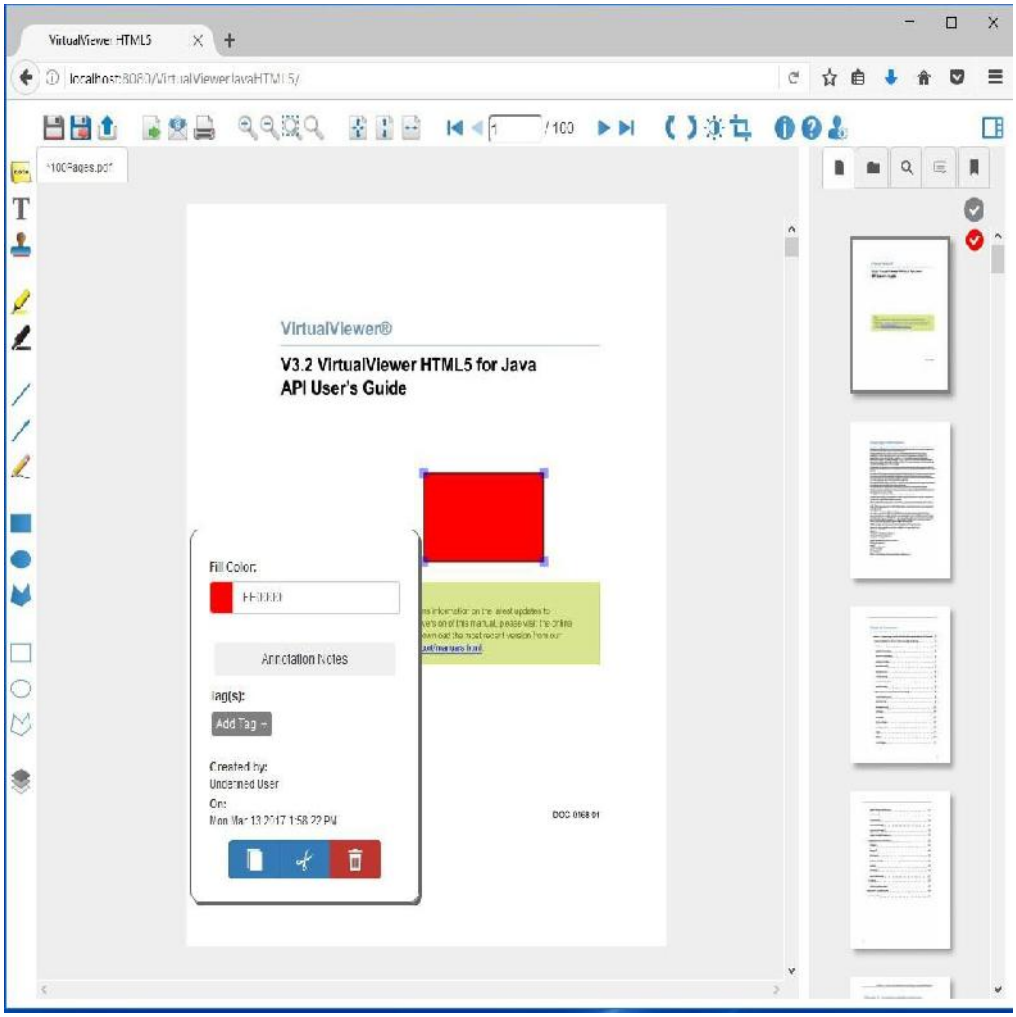


To adjust the line size, right-click on the line annotation. In the contextual annotation box, select the **line size** from the available line weights of 1 to 9.

Copying and Pasting an Annotation

Follow the steps below to copy and paste annotations:

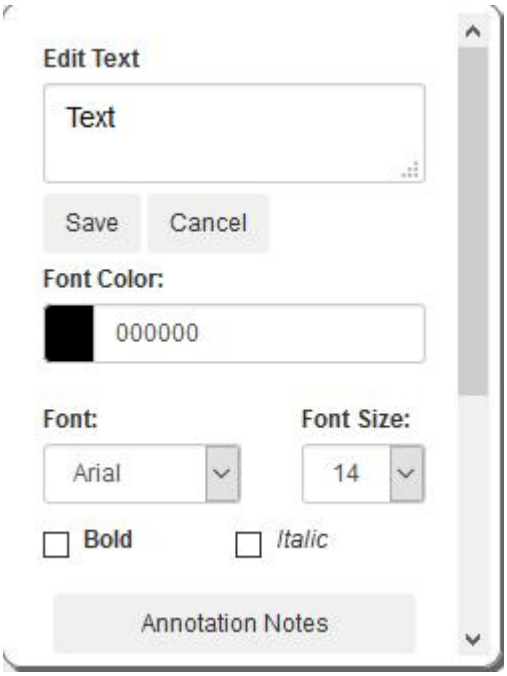
1. Right-click on an annotation.
2. From the dialog box, select the **Copy** or the **Cut** button.
3. Right-click on the page where you would like to paste the annotation. Select **Paste**.
4. The annotation is pasted on the page.



Editing a Sticky Note Annotation

Select the Sticky Note text annotation button.

The sticky Note Annotation appears as below. Enter your text. Select the check to save the text in the annotation. Select the X to delete the text last entered in the annotation.



To adjust the text color in a text annotation, select a **text color**.

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

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

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

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

Formatting changes are reflected in this text box as well as the text on the annotation. Select the Save button to save any text edit changes.

Moving an Annotation

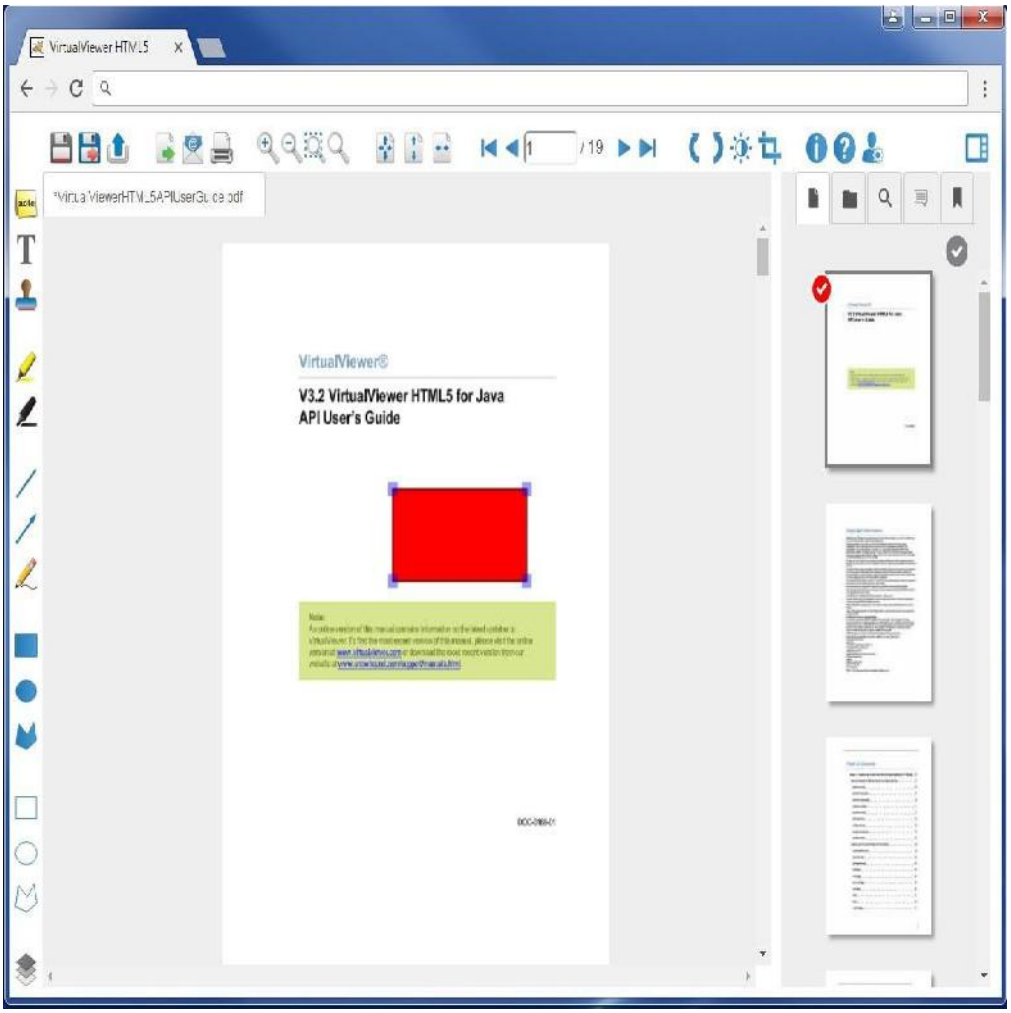
To move an annotation, click on it until it is highlighted and selection squares display on each of the annotation's corners. Drag the highlighted annotation until it is in the proper location.

Resizing an Annotation

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

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

Select any of the **other selection squares** other than the top left one to resize the annotation.



Annotation Indicators and Navigation

The annotation indicators and navigation buttons allow you to navigate through a document showing only the annotated pages.

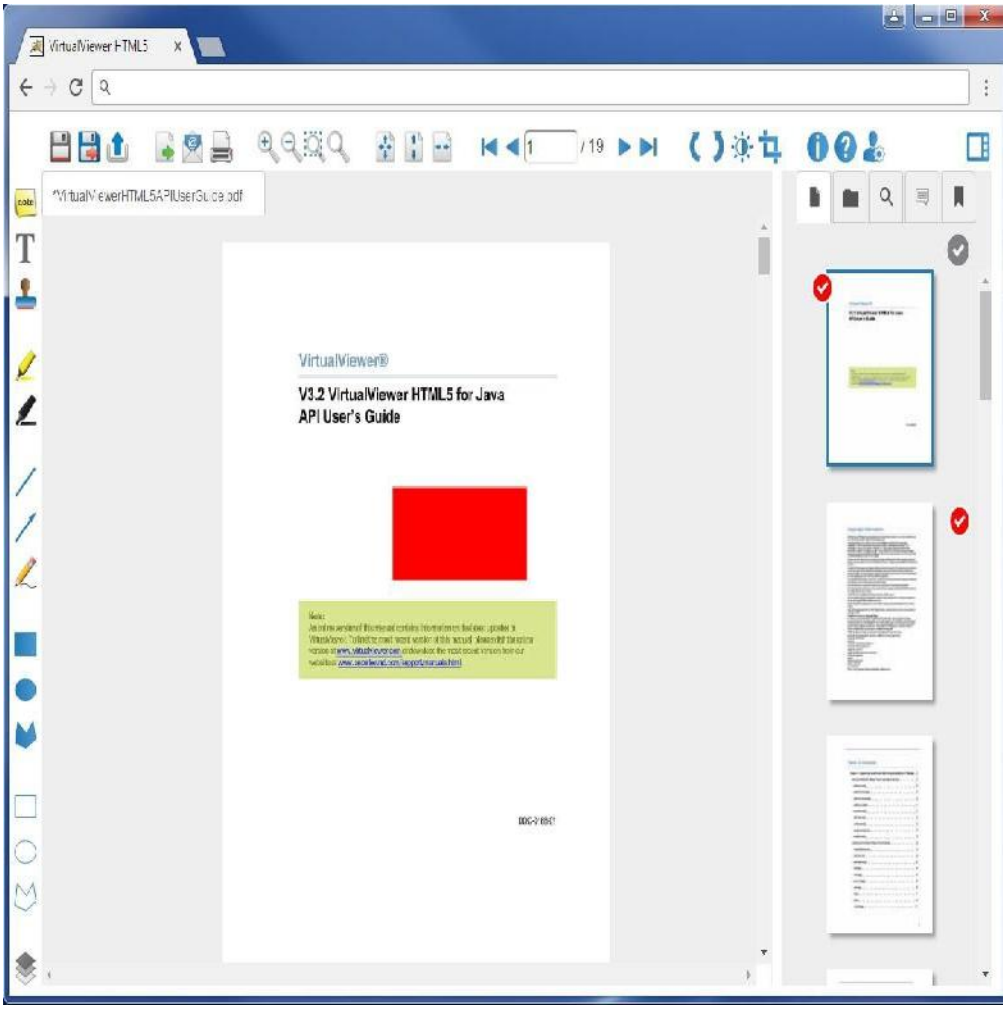
Click the Toggle Annotation Navigation button  (the grey checkmark on top right corner of the Pages panel) to display the Annotation panel.

The Annotation panel shows the **Previous Annotation Page** button, the **Next Annotation Page** button, and the **Filter** button.

Use the Next Annotation Page button and the Previous Annotation Page button to navigate from annotated page to annotated page skipping pages with no annotations.

Select the **Filter** button to clear the pages view in the thumbnail panel and only display pages that contain an annotation.

The Annotation Indicator icon displays as a red check in the upper right corner of a page with an annotation.



The annotation navigation buttons are enabled by setting the `showAnnNavToggle` config.js parameter to true. The default value is false. Please see the example below:

```
showAnnNavToggle: true,
```

The annotation indicator is enabled by setting the `showAnnIndicators` config.js parameter to true. The default value is false. Please see the example below:

```
showAnnIndicators: true,
```

Saving Annotations

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

Deleting Annotations

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

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

Revision history for Annotation Create Date/Time

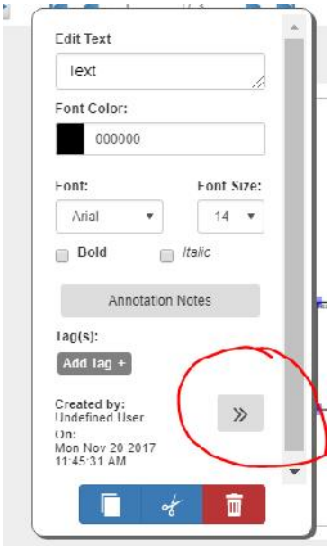
Use Case:
User 1 creates a sticky note on page 1 of a document, the userId/date/time are recorded
User 2 edits that same sticky note (color, size, placement, any change really. .. ect)
The userId/date/time are updated in a scrolling list reflecting each edit to that object.

Works for all annotations, image stamps and redactions
Resizing/moving records as a change
Changing text records as a change
Page manipulations or moving pages with annotations to a new/other document will not record a change

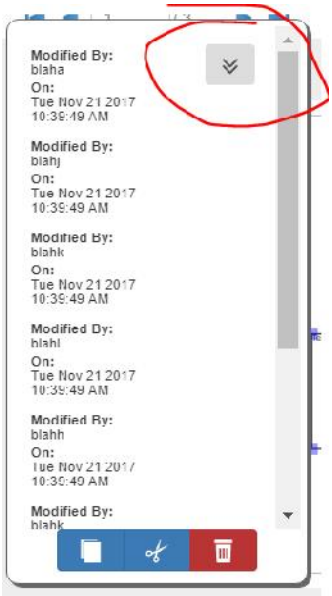
The use of pages with annotations in a VD will record changes to the annotations
A modification will be added if the annotation is modified and then saved.
If the user changes the color, moves around the annotation, and expands the annotation, and then saves once, only one modification item will be saved.
If an annotation is pasted, it will have a clean slate--it won't keep the modifications of the original.
If a document is saved-as, the annotations on the new document will still have a modification trail but will not add a new modification for the saving event.

Display and Use

The revision history is displayed in the annotation popup by clicking an expando button:

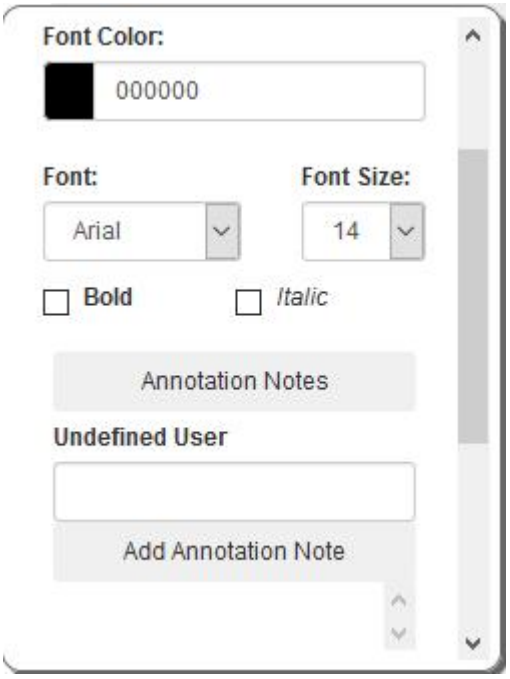


And shrunk again by hitting the same button:



Annotation Information

To display the **User Name** and **Date and Time**, right-click on any annotation in VirtualViewer. The annotation creator's user name and the date and time that the annotation was created display at the bottom of the annotation window.



Using Text Edit Annotations

Text Edit 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 Text Rubber Stamp Annotations.

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

T

You can dynamically resize text annotations. The text annotation box expands horizontally as you type from left to right.

In **config.js**, set the `autoResizeTextAnnotations` parameter to **true** to dynamically resize annotations. The default value is false.

Example:

```
autoResizeTextAnnotations: true,
```

If the `autoResizeTextAnnotations` parameter is set to true, the text annotation will act as follows:

The text annotation automatically resizes to fit the initial text when it is created.

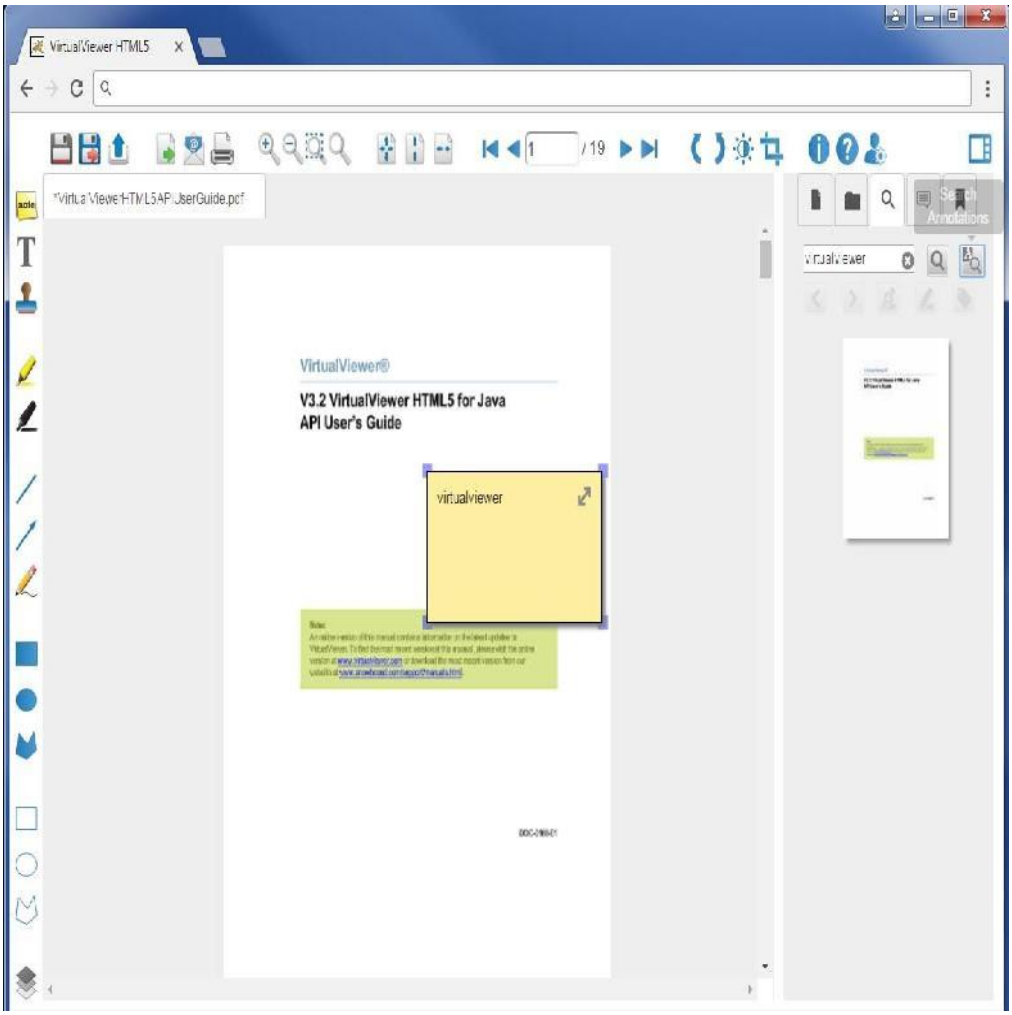
The text annotation extends the right edge of the annotation edit box as you type.

The text annotation is resized vertically and does not extend beyond the bottom of the page.

The horizontal resizing ends at the end of the page.

Search Annotation Text

To search annotation text, select the **Annotation Search** button in the Search tab in the Thumbnail panel. Enter the text that you are searching for in the Search field and select enter. The annotation text is highlighted for the term that you searched.



Using Image Rubber Stamp Annotations

An Image Rubber Stamp is an image annotation from a pre-defined list. Your system administrator has the ability to define a list of pre-configured Image Rubber Stamps through the `customImageRubberStamps` parameter in the web.xml file. For more information on configuring rubber stamp annotation functionality, please see Configuring Image Rubber Stamp Annotations.

The `customImageRubberStamps` parameter in the web.xml file specifies a comma-separated list of names which will be used to pull the individual stamp configurations out of the web.xml.

Click on the **Image Rubber Stamp** annotation toolbar button as shown below to see a list of available image rubber stamps defined by the system administrator.

Note:
This feature is not supported in Internet Explorer 8.
Image rubber stamps will auto size to a specific height and width.
Image rubber stamps behave the same as other annotation for permissions and layer settings.

Using Annotation Commenting

Annotation commenting allows you to add user comments to an annotation object. This allows multiple users to collaborate on a single annotation object. To use annotation commenting, follow the steps below:

User 1 creates an annotation or rubber stamp and saves and closes the document.

User 2 loads the document and right-clicks on the annotation or rubber stamp. The user enters text in the note field and selects **Add Annotation Note**. The note is displayed with the date and time that it was created.

User 3 follows the same steps as User 2. Each additional user can add comments.

To delete a note, select the x at the upper right of the note.

Annotation commenting is display only. Export, Send, Email and Print will not display the annotation comments on the pages.

In **config.js**, set the `enableAnnotationCommenting` parameter to true to enable annotation commenting. Set the parameter to false to disable annotation commenting.

To set the user name, select **User Preferences** and then select the **General Preferences** tab. In the **Username** field, enter the user name and select the **Save** button.

The Pages and Documents Panels

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

document made available by multiple documents mode. Page and document thumbnails now display the file name in a footer. This is managed by the displayThumbFooters parameter in the config.js.file. It is off by default.
To select a specific page or document simply click on the corresponding thumbnail and that page or document will load into the main viewing area.

Hiding the Pages and Documents Panel

The Thumbnail panel provides a convenient way to:

Navigate to any page in a document in the Pages panel.

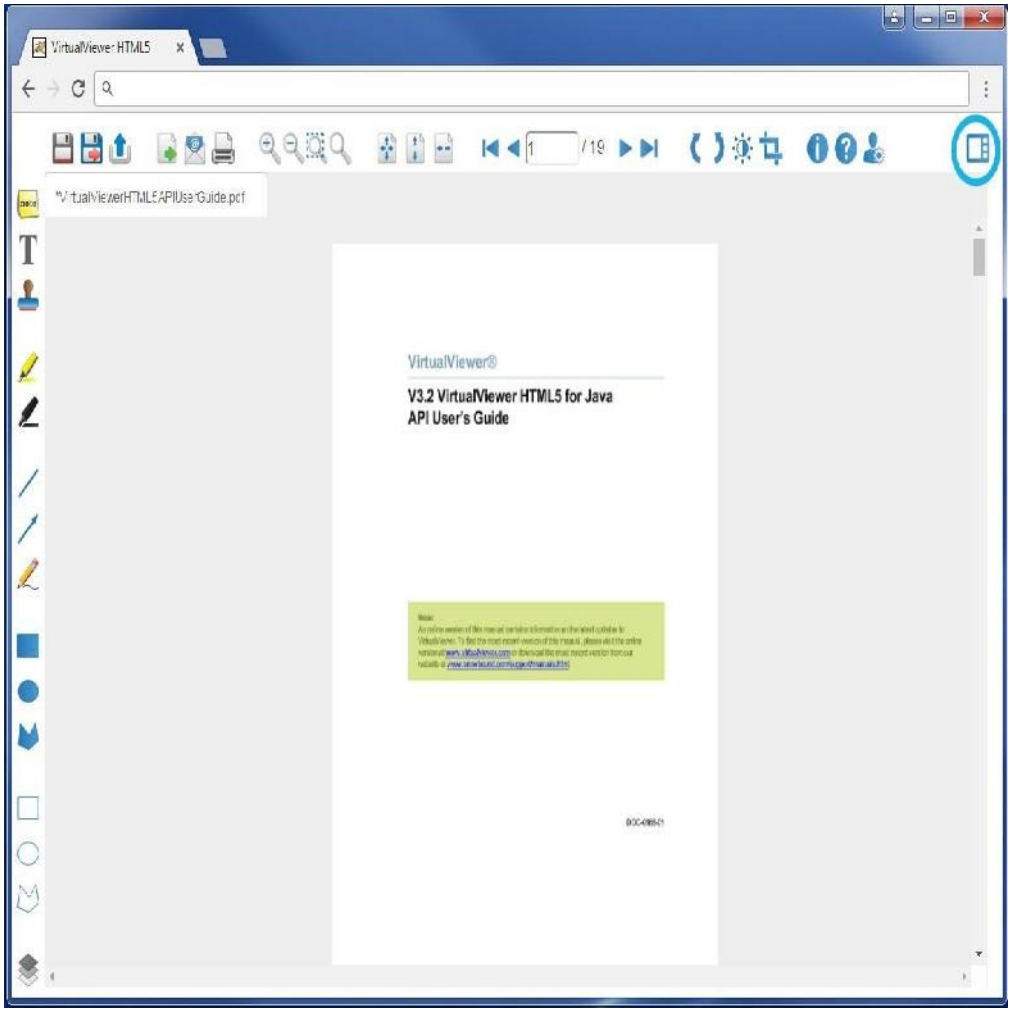
Select another document to view from the multiple Documents panel.

Create a new document by dragging and dropping pages from another document.

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

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

The following shows VirtualViewer HTML5 for .NET with the Thumbnail Panel hidden:



Split Screen View

You can launch documents to a lower panel to visually compare documents in one viewer session. The main image panel on the top retains all feature functionality. The lower panel includes all functionality except the magnifier and the thumbnail panel functionality including page manipulations, text search, document notes, and bookmarks.

Follow the steps below to use the Split Screen View feature:

1. On the **Documents tab**, right-click on the document thumbnail for the document that you want to open in the lower panel and select **Document Comparison**.

2. The document in the main image panel appears in the top panel. The document that you selected from the Documents tab appears in the lower panel.
3. Scroll to navigate the pages in the lower panel.
4. To replace the document in the lower panel, right-click on another document thumbnail in the Documents tab and select **Document Comparison**.
5. To undo the Split Screen View, right-click on the top panel, select **Undo Split**.

In **config.js**, set the `splitScreen` parameter to **true** to enable the Split Screen View feature. If the `splitScreen` parameter is set to false, the Split Screen View feature is disabled. The default value is true.

Example:

```
splitScreen: true,
```

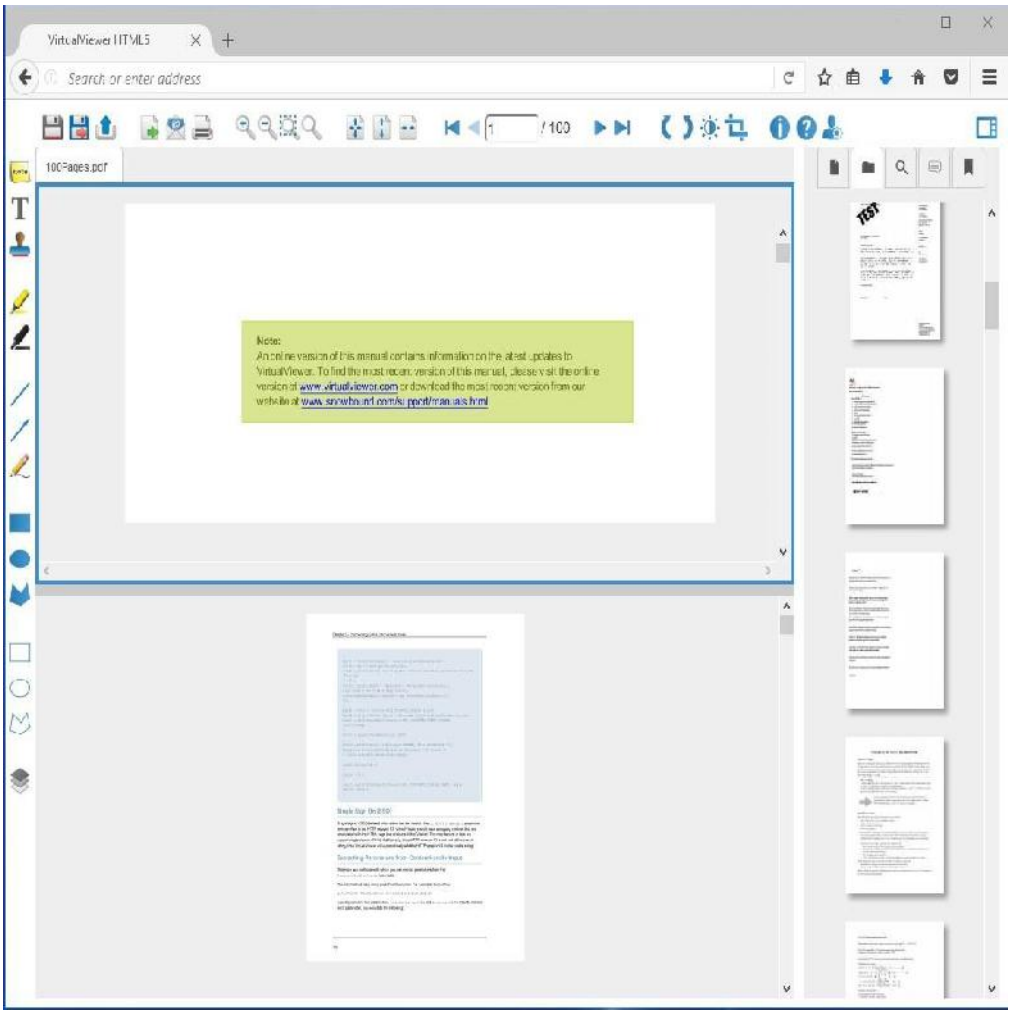
In **config.js**, set the `screenSizes` parameter to the screen size for panel 1 and panel 2. The first value sets the size of screen panel 1. The second value sets the size of screen panel 2. If the first value is set to 50, the first screen panel is set to 50% of the viewer. If the second value is set 50, the second screen panel is set to 50% of the viewer.

Example:

```
screenSizes : [ 50, 50],
```

As images are added by selecting **Split Image**, each new document request replaces the existing document in the lower panel.

The following shows the Split Image view:



The following sections describe the Document Notes, bookmarks, text searching, redaction, and page manipulation features.

Document Notes

The **Document Notes Panel** allows you to add notes that are relevant to the active document that you are currently working with. It includes the ability to view, create, edit, and delete notes.

You can configure the ability to only add a note by setting the `abilityToAddNotes(bool)` function in **webviewer.js** in the js directory to the following:

If bool == true, then you can add a note.

If bool == false, then you cannot add a note.

You can configure the ability to only edit or delete a note by setting the `abilityToModifyNotes(bool)` function in **webviewer.js** in the js directory to the following:

If bool == true, then you can edit or delete a note

If bool == false, then you cannot edit or delete a note

The `getDocumentNotes(string)` function in **webviewer.js** in the js directory will change the note's author to whatever name is specified in the string.The string will replace "User Unknown" with whatever string is entered in this function.

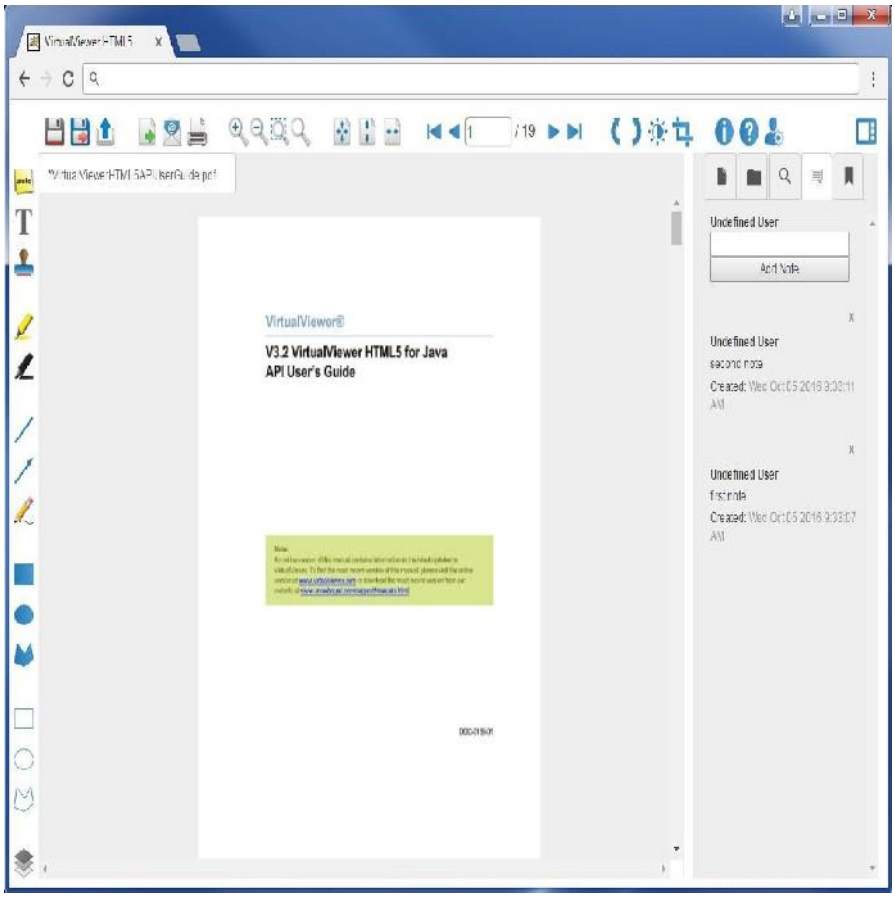
To set the user name in the Document Notes panel, set:

```
virtualViewer.getDocumentNotes(" ")  ,
```

For example, if you want to set the user name as Fred:

```
virtualViewer.getDocumentNotes("Fred") ,
```

The time stamp is set by the server time for the computer of the user who created the note. The time stamp changes for the server time for the computer of the user when edited.



Creating Document Notes

To create a note, follow the steps below:

1. Select the **Notes Tab**.
2. In the Document Notes field, add the text for the note.
3. Select the **Add Note** to add the note.

Document Notes templates

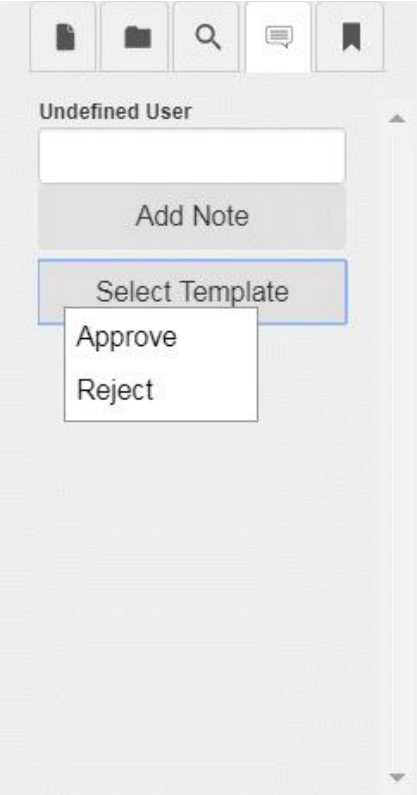
The ability to create Document Notes templates was added. The user can create a Document Notes template in two ways, either by adding the Document Note templates in **User Preference > Notes Templates** tab or by adding the template objects to the “noteTemplates” in **config.js**.

Follow the steps below to add a Document Notes template in User Preference:

1. Select the **User Preference** button**.** From User Preference dialog box, choose **Notes Templates** tab.
2. Select **Add** button to create new Document Notes template.
3. In **Template Name** field, enter the **template name**.
4. In **Template Text** filed, enter the **template text.**
5. Select **Save** button to save the template
6. To edit the Notes template, select the template then edit the **Template name** or **Template Text** field. Select **Save** button to save update template or **Cancel** button to exit.
7. To delete a template, select the template then select **Remove** button. Select **Save** button to save change or **Cancel** button to exit.

Document Notes Template workflow:

1. Select **Notes Tab**
2. Left-click **Select Template** button
3. Choose a template from the template drop down menu
4. Select **Add Note** button to add template to the document note.



Editing Document Notes

To edit a note, follow the steps below:

1. Double-click on a previously created note text to edit it.
2. In the Document Notes field, edit the note.
3. Select the **Apply** button to save the changes to the note.

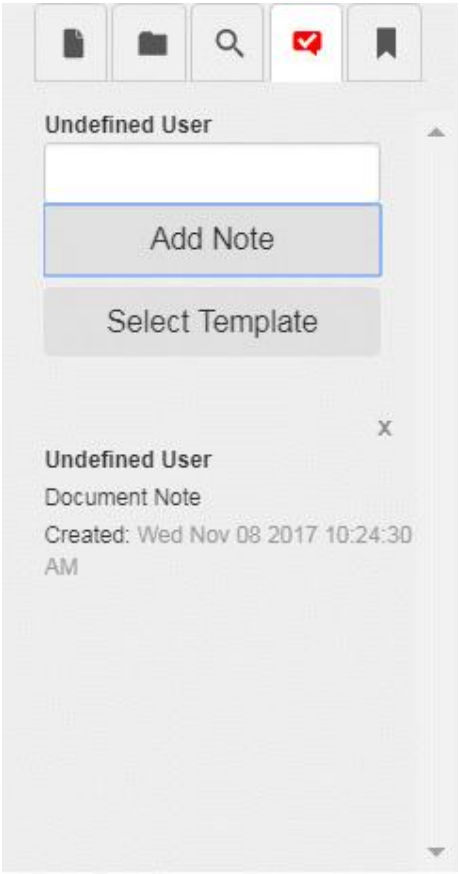
Printing Document Notes

To print a note, follow the steps below:

1. Select the **Print** button.
2. Check the **Print Document Notes** check box and select the **Print** button. The "Save as PDF" option was removed because it has become a redundant feature that is now handled with the "Export" feature.
3. The Document Notes print.
4. The Document Notes print on the last page of the document.

Document Notes Indicator

A red checkmark Document Notes indicator on the **Notes Tab** will toggle on if a document contains a document note, otherwise it will be toggled off.



Watermark Support

VirtualViewer now offers watermarks for customers who need to mark page backgrounds with specific notifications such as “Private”, “Confidential”, “Do not distribute” and so on. Watermarks can be created that are transparent or solid, of varying fonts and sizes and positions. They can also be restricted to admins versus all users.

They can also have dynamic tags for user name, page numbers, print time, and document name.

What does the User Interface look like?
There are a few UI changes. A new watermarks dialog lays out all the watermarks options for creation, deletion and editing. In the "document handling" dialogs (printing, exporting, etc) a new checkbox has been added, so the user may decide whether to burn their watermarks when exporting.

If a watermark is marked as admin-created, then the "burn watermarks" option will be checked and disabled, so the admin watermarks burn by default. Similarly, a non-admin may not edit or delete admin-created watermarks.

Those dialogs are the only way to interact with watermarks. You can’t select them on the document, move them around, etc--they’re not annotations, they’re marked into the document once created.

Other features
Users may add dynamic data into their watermark text. This is easily done in the watermarks dialog by clicking on a tag button above the text box in the watermark dialog. If you inspect the raw text of the JSON, a tag will appear enclosed in two @ signs, which may be escaped by adding a /. When displayed, the tag will be replaced by data.

For instance, the user wants a page number to print on each page. They click the tag button in the dialog. In JSON, now the watermark text would say, "Page @@pagenumber@@". When displayed on the document, the watermark on page one will read "Page 1", the watermark on page fifty will read "Page 50" and so on. If the user types "If I wanted a page number I would use /@/@pageNumber/@/@", the watermark will now display "If I wanted a page number I would use @@pagenumber@@". The tag is escaped, and so is not replaced by a dynamic number.

Available tags are:

- Username: the user’s username as stored in user preferences.
- total pages: the number of pages in the document.

- current page number: the number of the current page.
- print time: The date & time when the document was exported or printed. When displaying in the viewer, this is just an example date and time, from when the document was opened.
- document name: The display name of the current document.

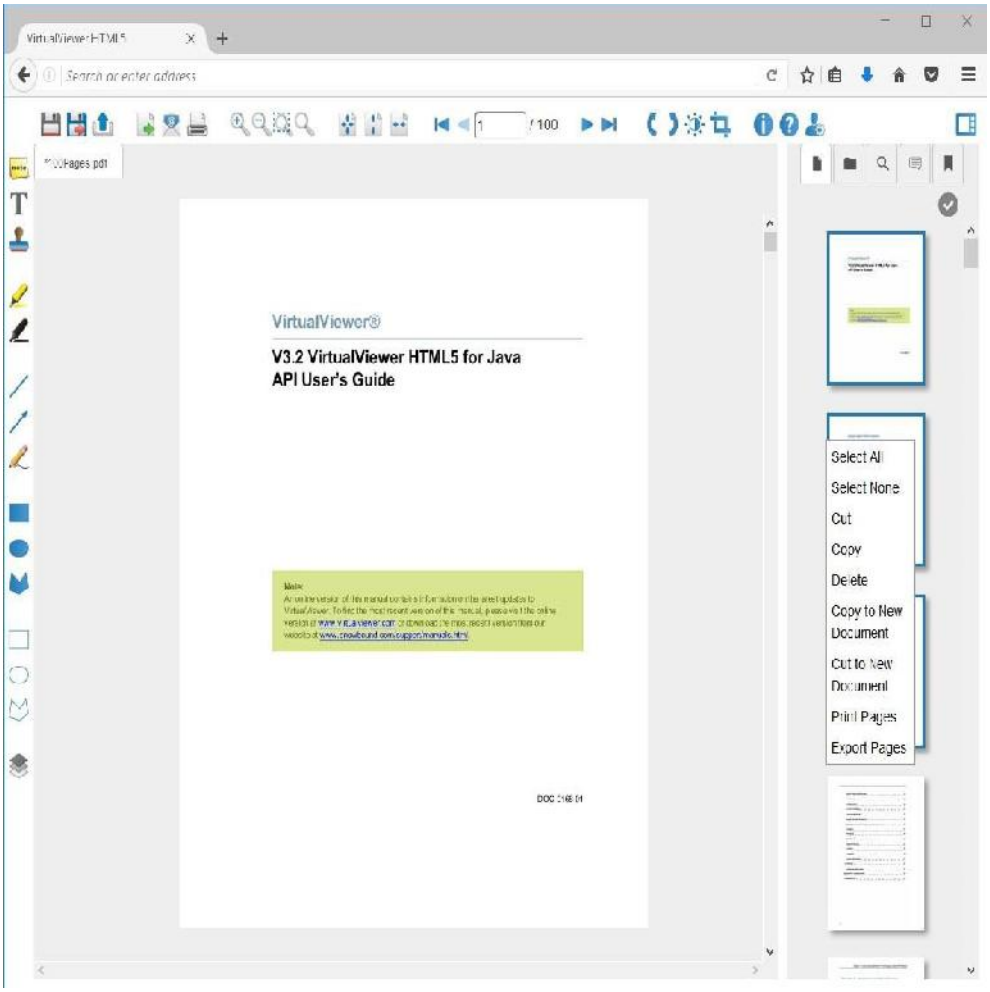
Select Pages from the Thumbnails Panel

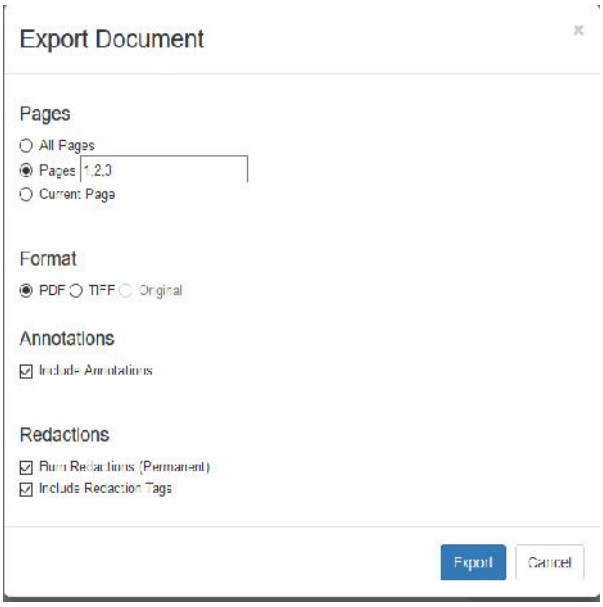
You can select pages from the Thumbnails panel for Export Pages or Print Page.

Follow the steps below to select pages from the Thumbnails panel for Export Pages or Print Pages:

1. Select one or multiple thumbnails from the Thumbnails panel.
2. Right-click to see options for Export Pages or Print Pages.
3. Select **Export Pages** or **Print Pages**.
4. On the dialog box, select **All Pages**, **Pages** (enter the page range), or the **Current Page**. The dialog box automatically displays with the page range.

The following shows the Export Page or Print Pages option:


VirtualViewer®
V3.2 VirtualViewer HTML5 for Java
API User's Guide
Select All
Select None
Cut
Copy
Delete
Copy to New Document
Cut to New Document
Print Pages
Export Pages



Substitute Image Thumbnails

You can add a substitute box instead of image thumbnails. This improves performance because image thumbnails do not need to be created.

In **config.js**, set the `doNotLoadPageThumbs` parameter to **true** to display substitute boxes instead of image thumbnails. The default value is false.

Example:

```
doNotLoadPageThumbs: true,
```

If the `doNotLoadPageThumbs` parameter is set to **true**, VirtualViewer will not request thumbnail images. Instead, VirtualViewer displays a box with the page number. Select the substitute thumbnail box as you would an image thumbnail.

Use the `thumbPageLabel` string to set the page thumbnail tooltip. Please note that it is important to include the trailing space in "Page ".

Example:

```
"thumbPageLabel": "Page "
```

Extract and Append Page Ranges

You can extract and append a range of pages instead of the entire document when saving to PDF. All pages do not have to be processed during saving.

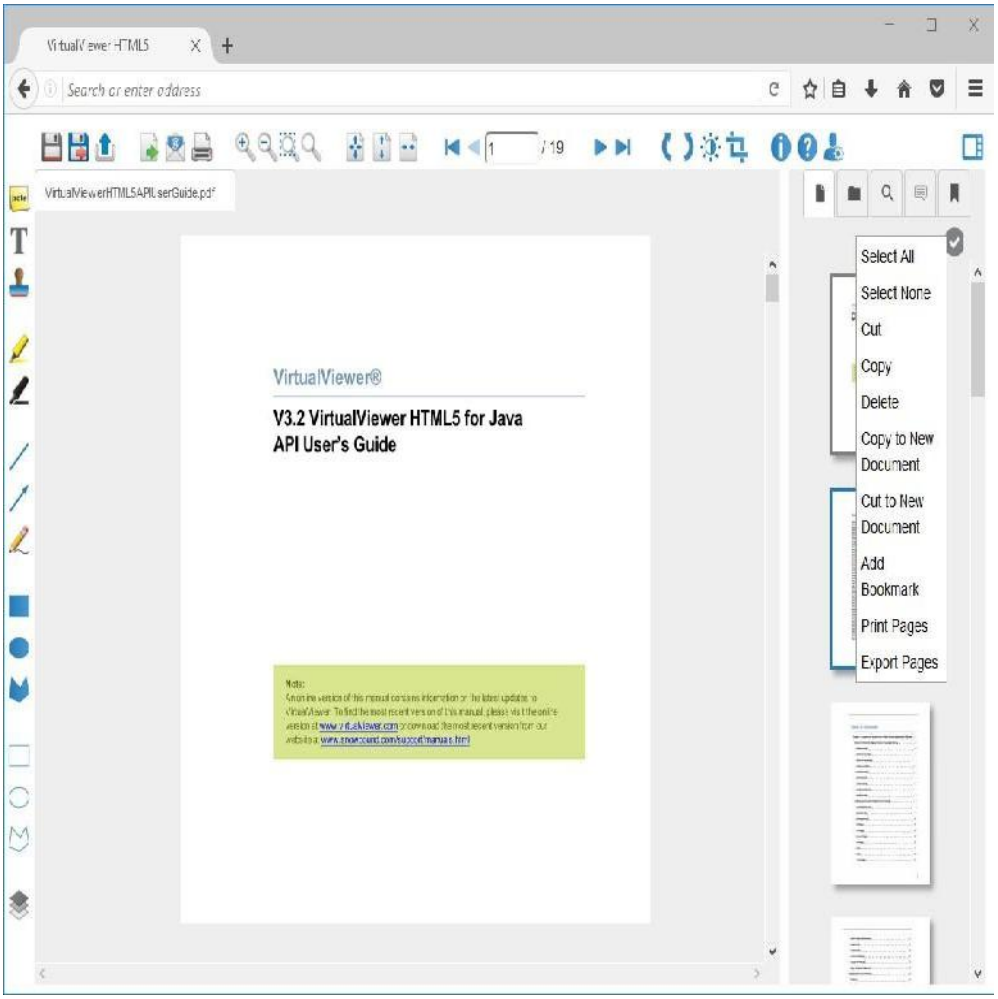
This provides a shorter save time for documents with a large number of pages (100+).

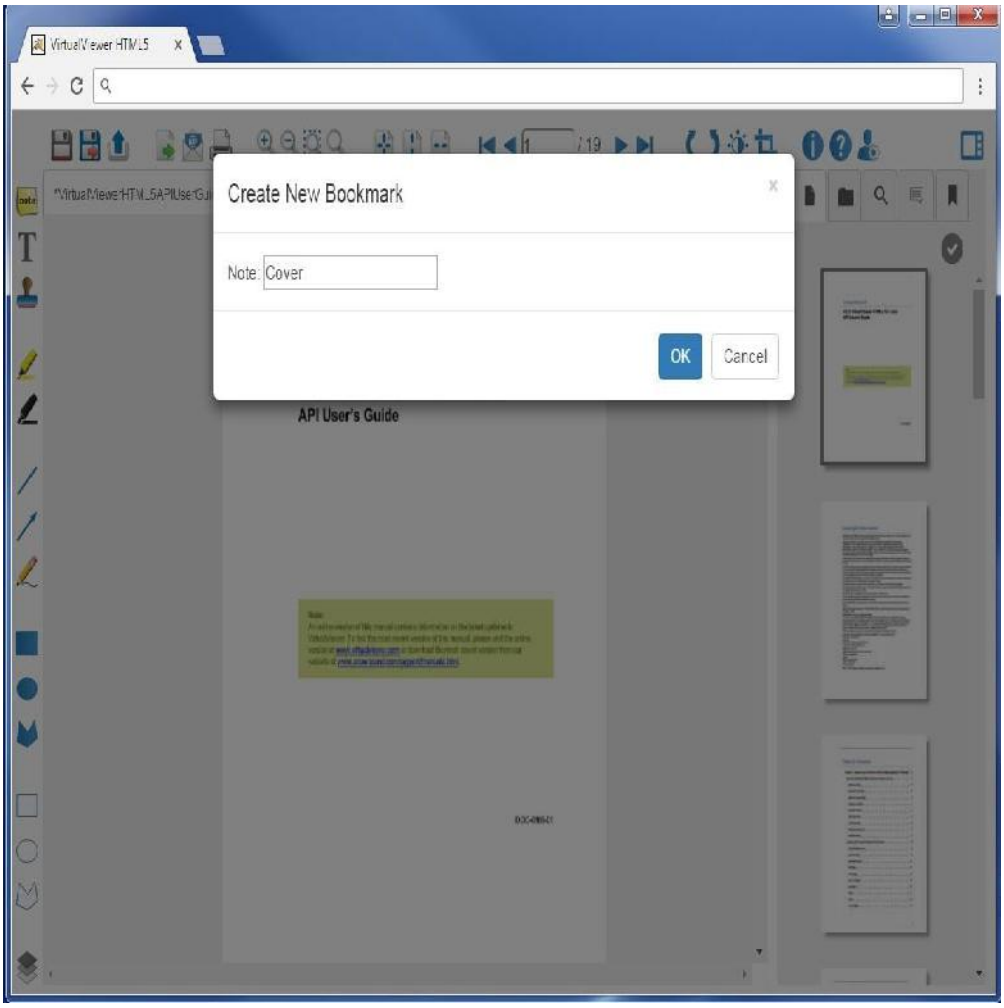
Bookmarks

The **Bookmarks Panel** allows you to add bookmarks that are relevant to the active document that you are currently working with.

Creating Bookmarks

To create a bookmark, right-click on the Pages Tab and select **Add Bookmark**. In the Create a New Bookmark dialog, add the text for the bookmark and select **OK**.

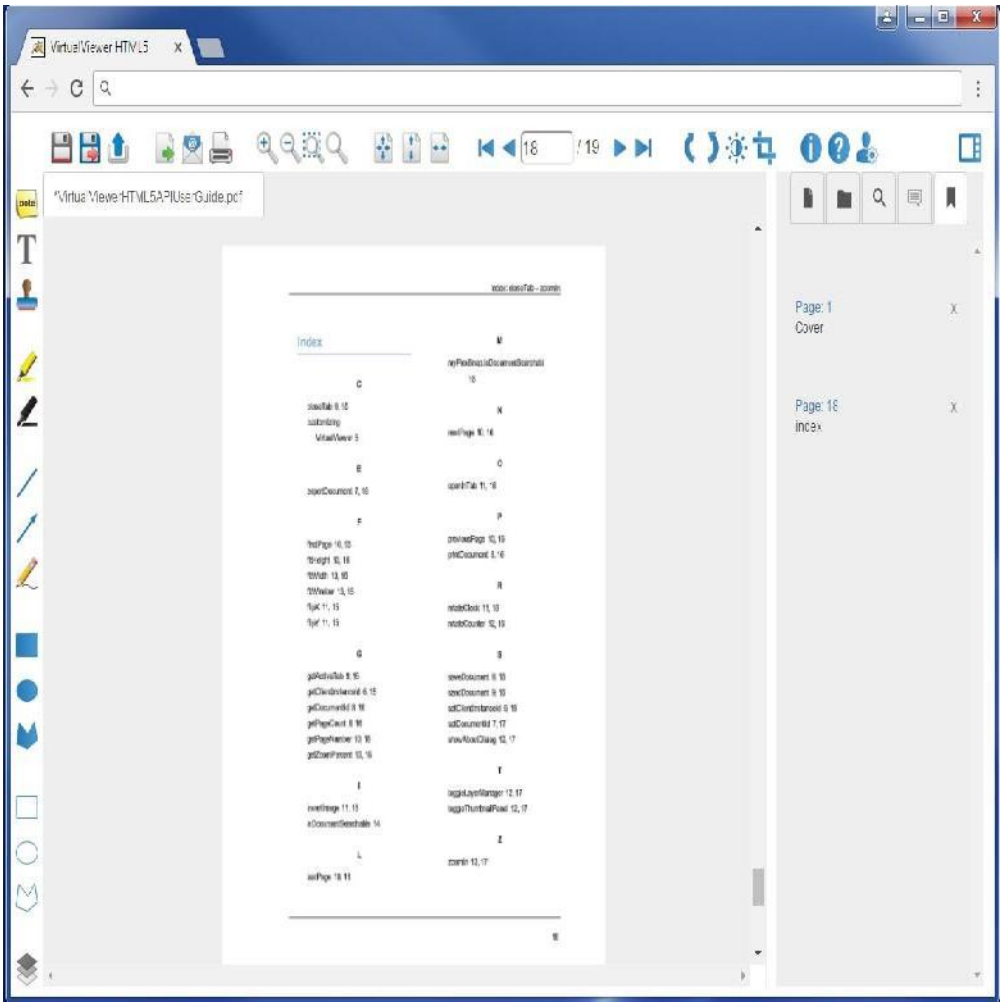




Viewing Bookmarks

To view bookmarks, select the Bookmarks tab. The Bookmarks tab displays a list of all bookmarks created in that document. The list of bookmarks display the page number and text entered by the user. For example:

Page. 39
This is the signature page.



The bookmarks feature is not supported in Virtual Documents, Print, Export, Save Document As, and Page Manipulation.

Text Searching

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

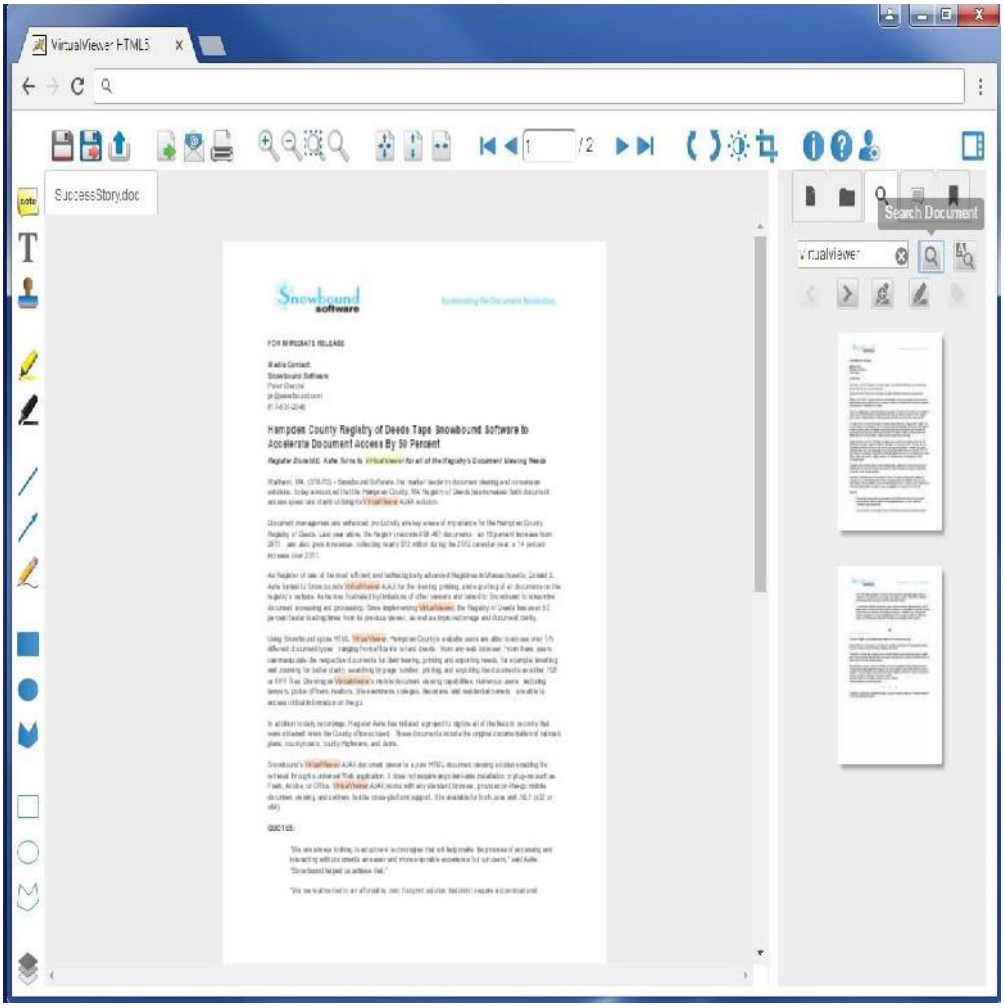
Example 1.6: Enabling TextSearching

```
var showSearch= true;
```

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

Example 1.7: Enabling Case Sensitivity in Text Searches

```
var searchCaseSensitive = false;
```



Setting the Default Colors

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

Pattern Based Text Searching

You can search for patterns in text including social security numbers, phone numbers, credit card numbers, and email addresses. You can use this information to quickly locate, redact, or collaborate on important information within documents.

To search pattern based text, follow the steps below:

1. From the **Search** tab, select from the drop down for the available patterns to search. For example, select a **social security** pattern. VirtualViewer searches the document for all patterns matching a social security pattern: ###-###-####.
2. VirtualViewer highlights patterns returned by this search as it highlights any text search results.
3. Use navigation arrows to scroll through the patterns results.
4. Use the **Redact Current Match** button to redact the current pattern search result. Use the **Redact All Matches** button to redact all pattern results.

The available patterns include:

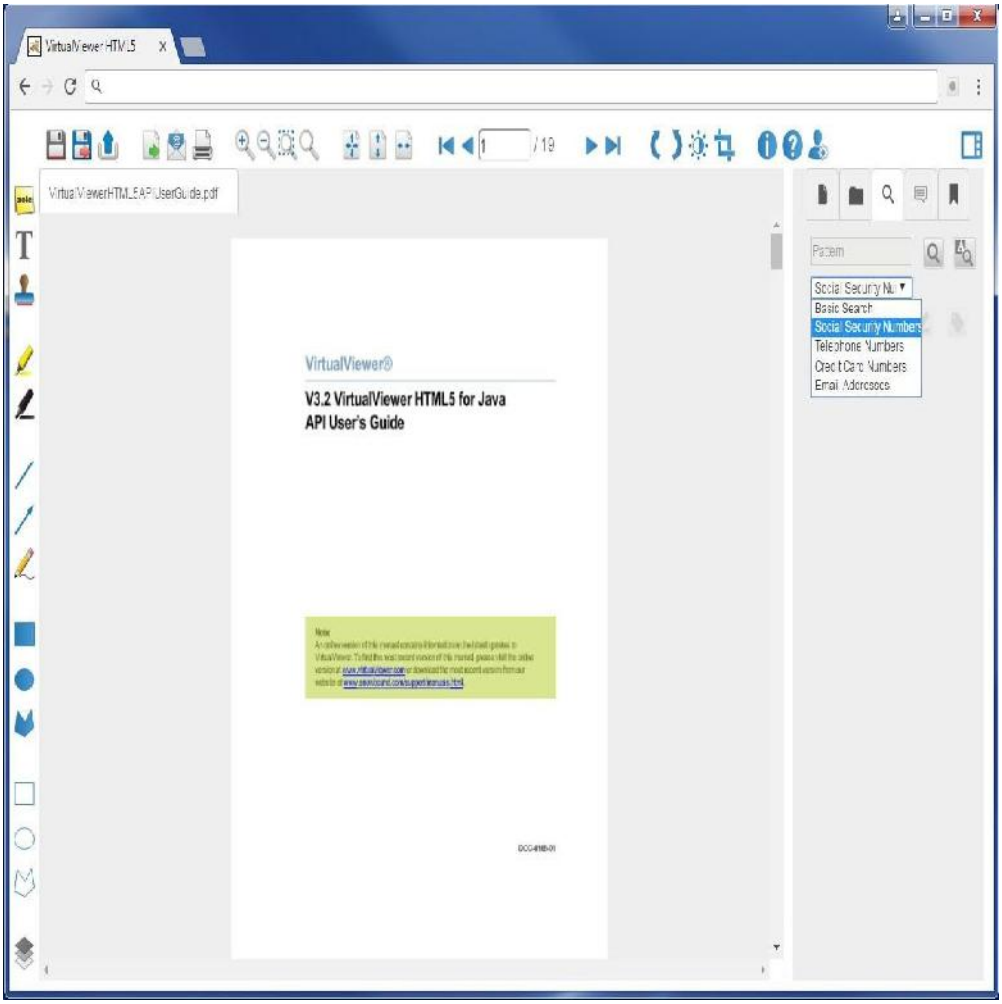
Social Security Number:
123456789
123-45-6789
123 45 6789

Telephone Number:
6176072000
617 607 2000
617-607-2000
(617)-607-2000
617.607.2000
(617).607.2000

Credit Card Number:
1234567891234567|
1234-5678-9123-4567

1234 5678 9123 4567
1234.5678.9123.4567
123456789123456
1234-567891-23456
1234 567891 23456
1234.567891.23456

Email Address:
Any string including a @ symbol with characters on either side of the symbol.



Additional Notes

Pattern based text searching works with any format that is supported for searching text. This includes AFP, PCL, PDF, Word and Excel.

A text pattern search result that breaks on two separate lines will not be found..

A text pattern search that contains odd text spacing between characters may not be found.

Working with Redactions

This section explains the redaction feature and how to work with redactions.

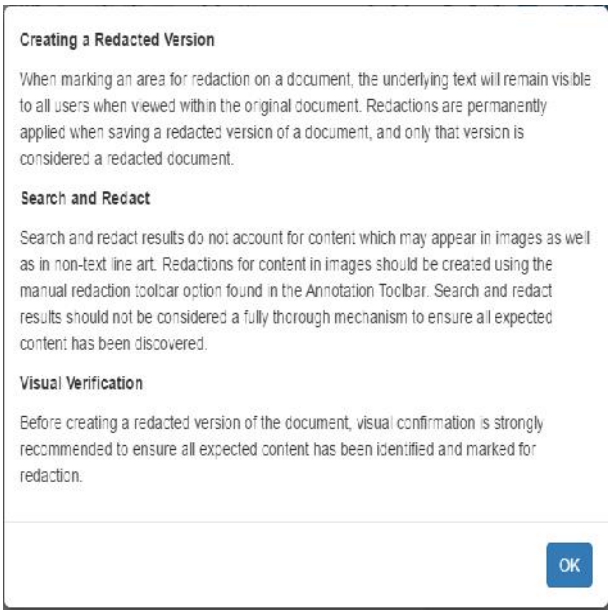
It is important to note that any area marked for redaction will not be redacted until a new document is created from the original document by Save Document As or Export Document. The original document will only show the areas that were marked for redaction but those areas will not be permanently redacted until a new document is saved through Save Document As or Export Document.

There are three ways to mark an area for redaction:

1. Select the Redaction Area tool from the Annotation tool bar.
2. Select text, right click and then select Redact from the menu.
3. Search and redact: You can step through the search results and mark each redaction by selecting the **Redact Current Match** button or select the **Redact All Matches** button to redact all search results.

Redaction Information Warning Message

A **Redaction Information: Read Carefully** warning message displays the first time that you select the Redaction Area tool.



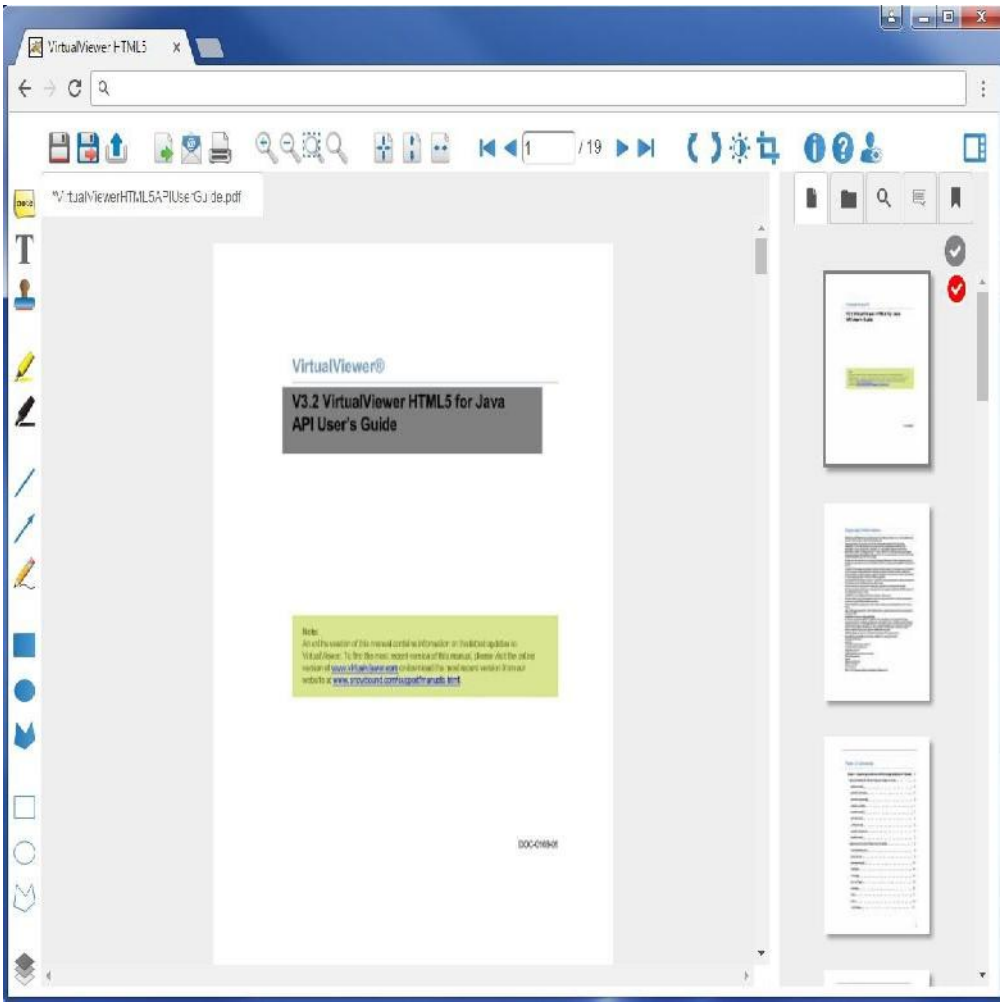
This message is triggered by any of the following events:

- You select the Redaction Area tool.
- You select Redaction from the selected text content menu.
- You select the Mark for Redaction button in Search and Redact.
- You open a document that contains saved Redaction annotations that have not been burned in.

Note:
This message shows once per user session when you initially work with a document with areas marked for redaction.

Redaction Areas

Select the Redaction Area button  from the Annotation toolbar to mark an area of your document for redaction. Drag your mouse to create translucent rectangle over the area that you want to mark for redaction. The redaction area is a rectangle with translucency. You can see through the rectangle and read the text behind the annotation.



To delete the redacted area, right-click on the translucent rectangle covering the text that you have marked for redaction and select the Delete button. In the Delete Annotation? dialog, select the Delete button.

Using the Redaction Area tool is the quickest way to mark an area for redaction.

Redacting by Selecting Text

Select the text that you want to redact. Right click and then select **Redact** from the menu.

To delete the redacted area, right-click on the translucent rectangle covering the text that you have marked for redaction and select the Delete button. In the Delete Annotation? dialog, select the Delete button.

Using this method to redact select text will only grab the vector text. (Embedded images cannot be redacted in this mode.) The advantage is that you can visually see the text selected for redaction and avoid marking the white space. The disadvantage is that each line is marked as a separate block marked for redaction. Each block needs to be edited or deleted separately.

Search and Redact

Use one of the following two methods for marking redaction areas in search results:

Clicking the **Redact All Matches** button applies redactions to all matches on all pages of the current document.

Clicking the **Redact Current Match** button moves to the next result and requires you to click the Redact button or skip the match and press the Next Match to move on to the next result.

Save Document As, Export Document, Email Document and Prints

The Save Document As, Export Document, Email Document and Print toolbar options include the Burn Redactions (Permanent) check box.

It is important to note that any area marked for redaction will not be redacted until a new document is created from the original document by Save Document As or Export Document. The original document will only show the areas that were marked for redaction but those areas will not be permanently redacted until a new document is saved through Save Document As or Export Document.

Page Manipulations with Redactions

When copying or cutting to an existing document, page manipulations will act as follows:

If pages that are copied/cut to an existing document contain redaction areas, those redaction areas will be copied to the new location, but not burned in.

When you select save, the page manipulations will be saved, leaving the redaction areas.

When copying or cutting to a new document, page manipulations will act as follows:

If pages with redactions are copied/cut to a new document, you will see the existing Dialog box to name the new document.

The new document is not saved until the user selects save.

When you save the newly created document, the document and the redaction areas will be saved.

Annotation Redaction Tagging

Annotation redaction tagging assigns a categorical value to individual annotations or redaction. The values are reasons why the annotation or redaction exists. For example, a social security number could be tagged with a Social Security Number value. Follow these steps to apply annotation redaction tagging:

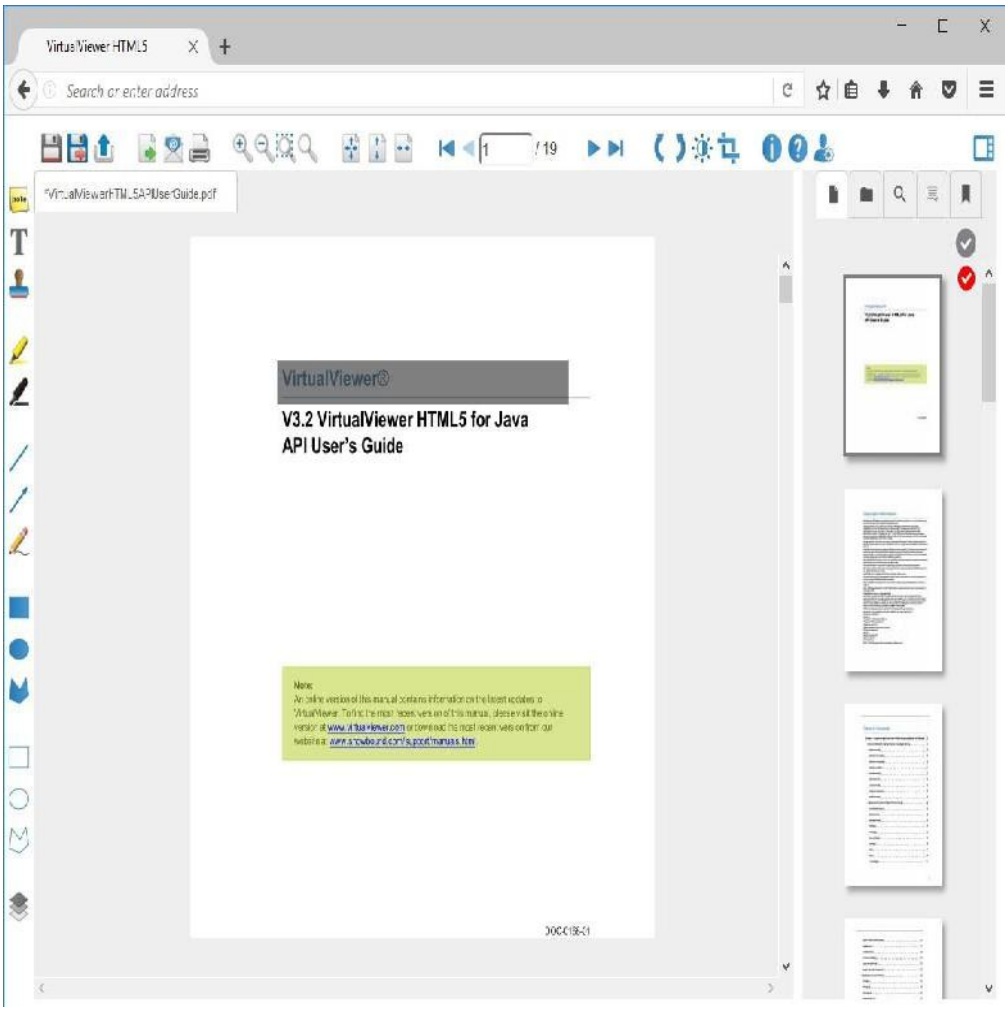
Draw a redaction object over the item that you wish to redact such as a social security number.

Right click on the redaction object when still in highlight mode and select from a predefined list of redaction tags. For Example, “Social Security Number.”

Select **Save Document As** to burn in the redaction with tag. The new document with the burned in redaction now has the Social Security tag to indicate it was a redacted social security number.

To configure your predefined list of annotation redaction tags, add the strings for your tags to the `annotationTags` array in the config.js file. Please see the example below:

```
annotationTags: ["Social Security Number", "Review"
],
```



Disabling Redaction Tags for Export, Print, Email, or Saving As

You can disable redaction tags when a document is exported, printed, emailed, or saved as.

Follow the steps below to disable redaction tags when a document is exported, printed, emailed, or saved as:

1. Create a redaction on your document. Right-click on the redaction, select **Add Tag** and select a redaction tag from the drop-down menu to add a redaction tag.
2. Select the **Include Redaction Tags** checkbox when selecting **Export**, **Print**, **Email**, or **Save Document As**

3. If you select the **Include Redaction Tags** checkbox, the redaction tags are included. If you uncheck the **Include Redaction Tags** check box, the redaction tags are not included.

The Include Redaction Tags checkbox defaults to checked. It is disabled if the Burn Redactions checkbox is unchecked.

Tag Results for Search and Redact

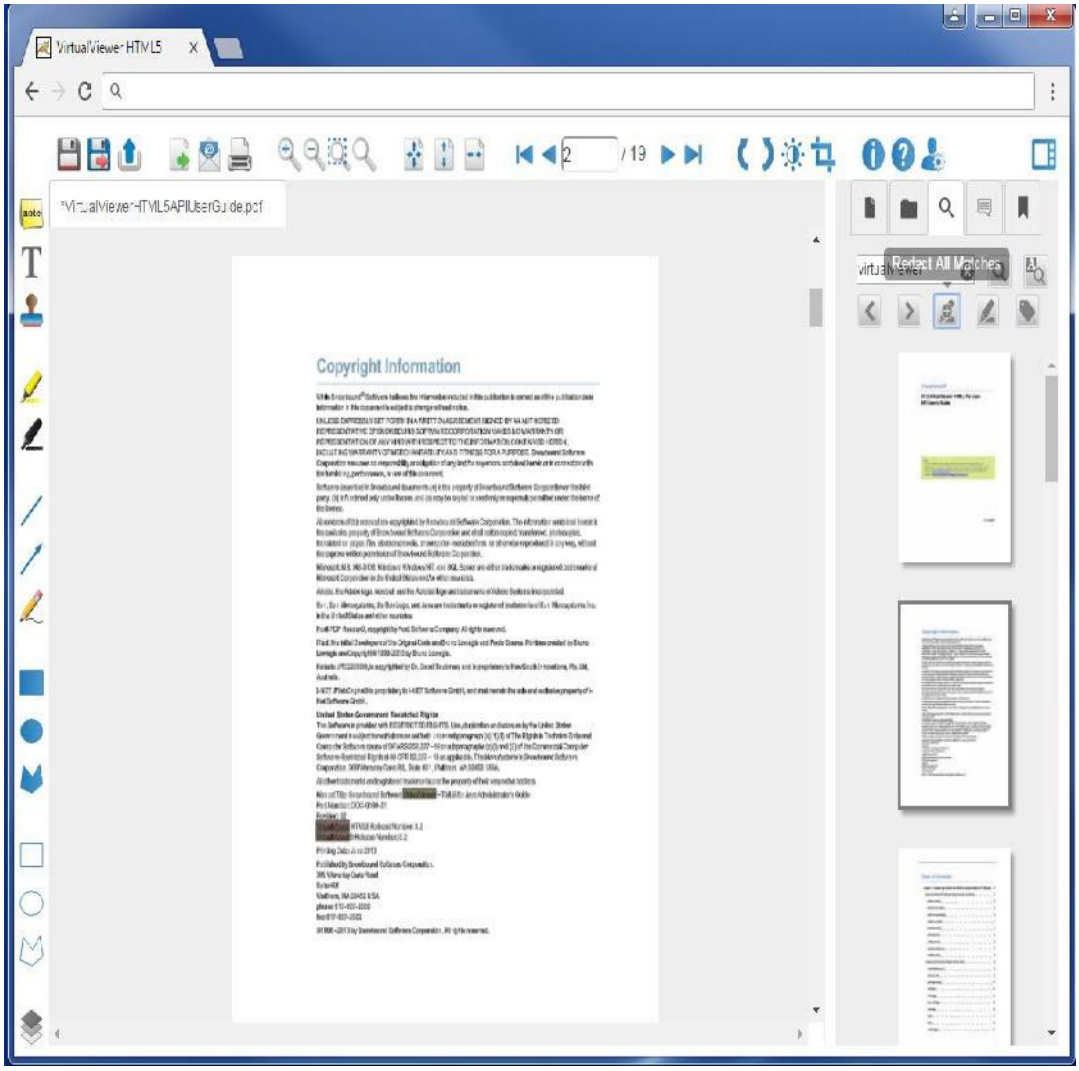
Follow the steps below to use the tag results for search and redact feature:

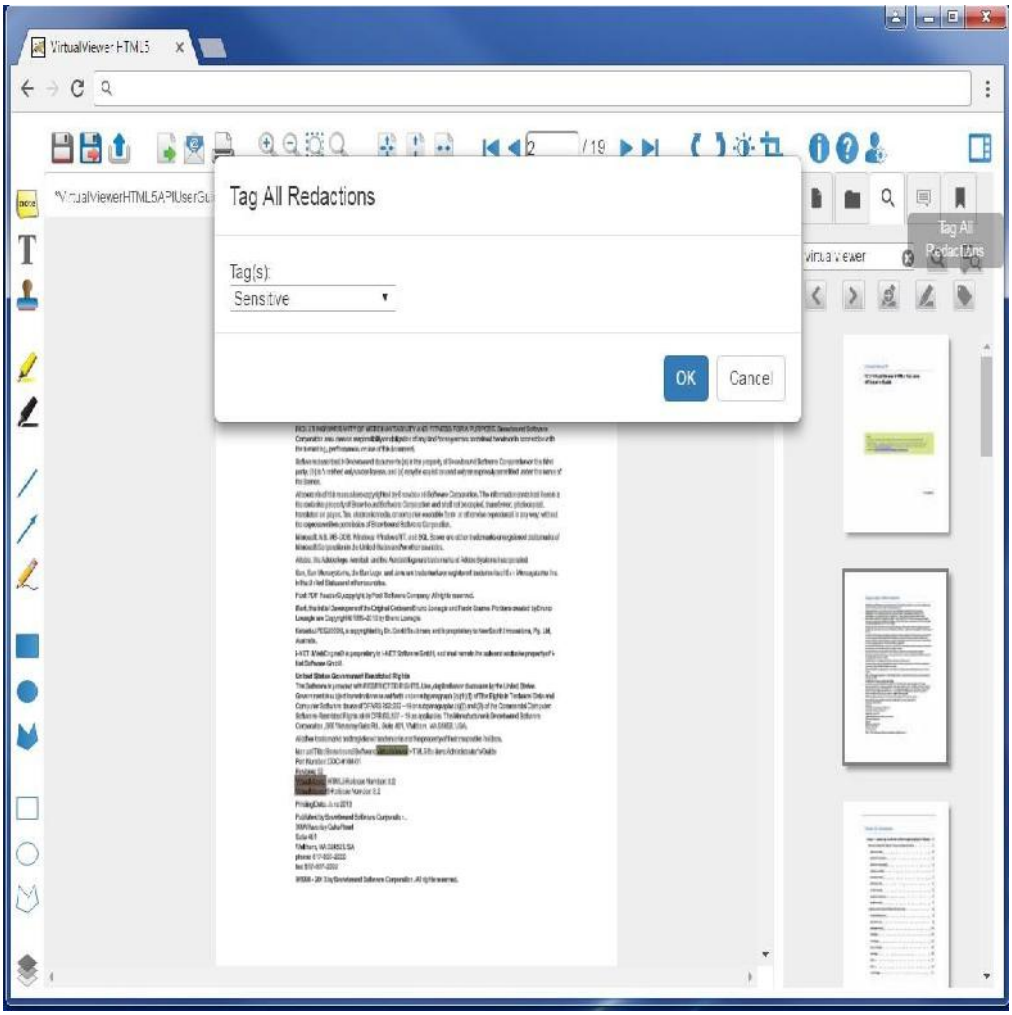
1. Load a text searchable document in VirtualViewer.
2. Search for a term. All results for that term are highlighted in the document.
3. Select **Redact All Matches**.
4. Select the **Tag All Redactions** button.
5. In the Tag All Redactions dialog, select the type of redaction from the Tag(s) drop down and select the *OK* button.
6. The redactions are now tagged.

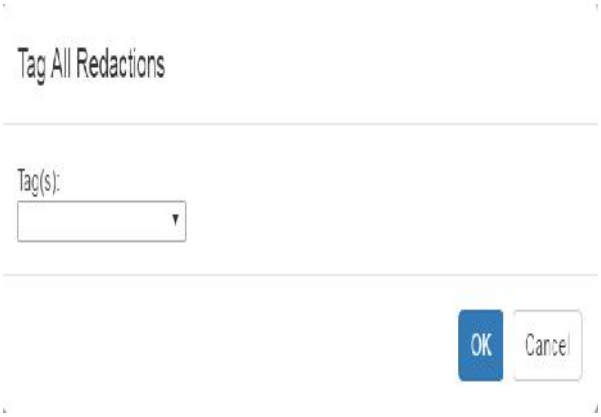
In **config.js**, set the `searchRedactionTags` parameter to true to turn on search redaction tags. The default value is true.

To configure your predefined list of annotation redaction tags, add the strings for your tags to the `annotationTags` array in the **config.js** file. Please see the example below:

```
annotationTags: ["Confidential","Redaction","Social Security","Credit Info"],
```


VirtualViewer HTML5
2
/ 19
Redact All Matches
Copyright Information


VirtualViewer HTML5
Tag All Redactions
Tag(s):
Sensitive
OK
Cancel


Tag All Redactions
Tag(s):
OK
Cancel

Page Manipulations

Manipulating Page Order using Thumbnails

VirtualViewer HTML5 for .NET 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 .

Selecting a Page

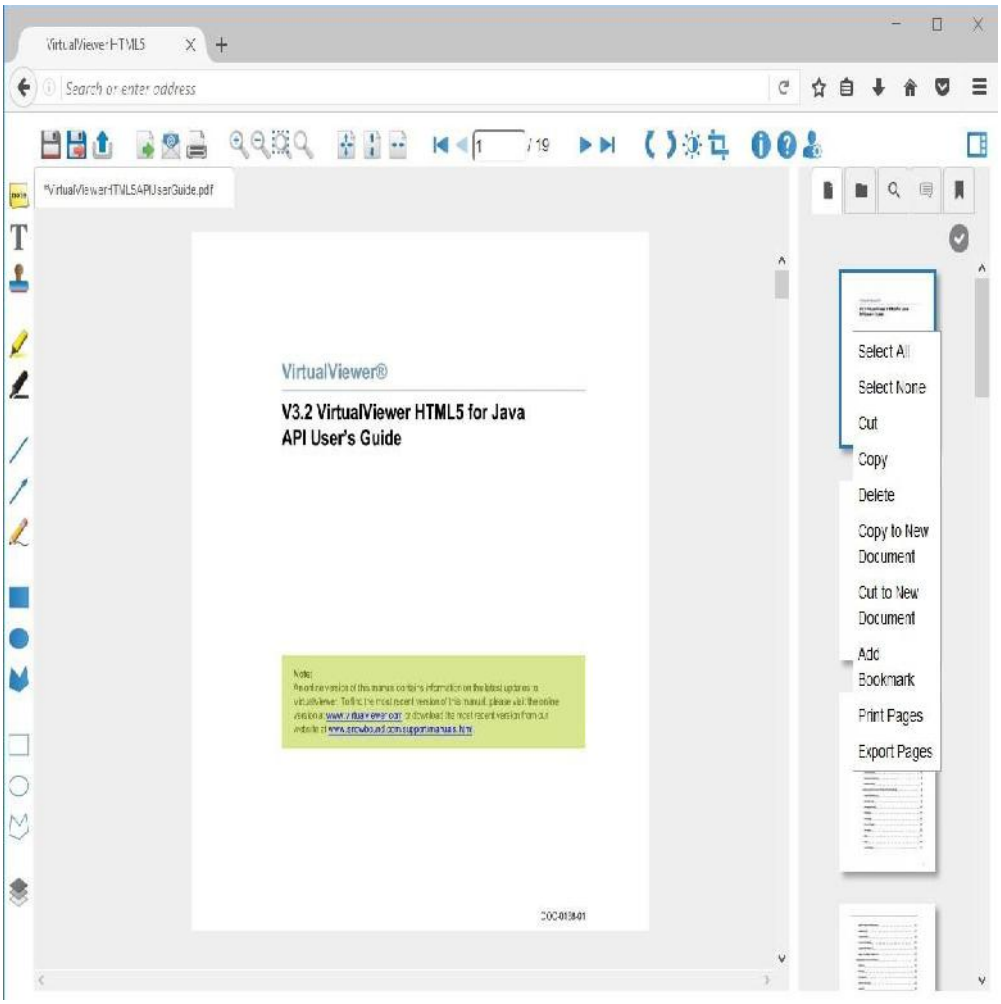
To select a page for page manipulation, left click on a page thumbnail in the Pages tab. A gray selection border around the thumbnail indicates that it has been selected for page manipulation.

Hold the **Ctrl** key while selecting multiple page thumbnails to allow the selection of all thumbnails selected for page manipulation.

Hold the **Shift** key and select a single thumbnail while one or more thumbnails are already selected to highlight all pages between the highest page selected before the new selection.

Loading the Page Manipulation Context Menu

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



Cutting, Copying, Deleting and Inserting Pages

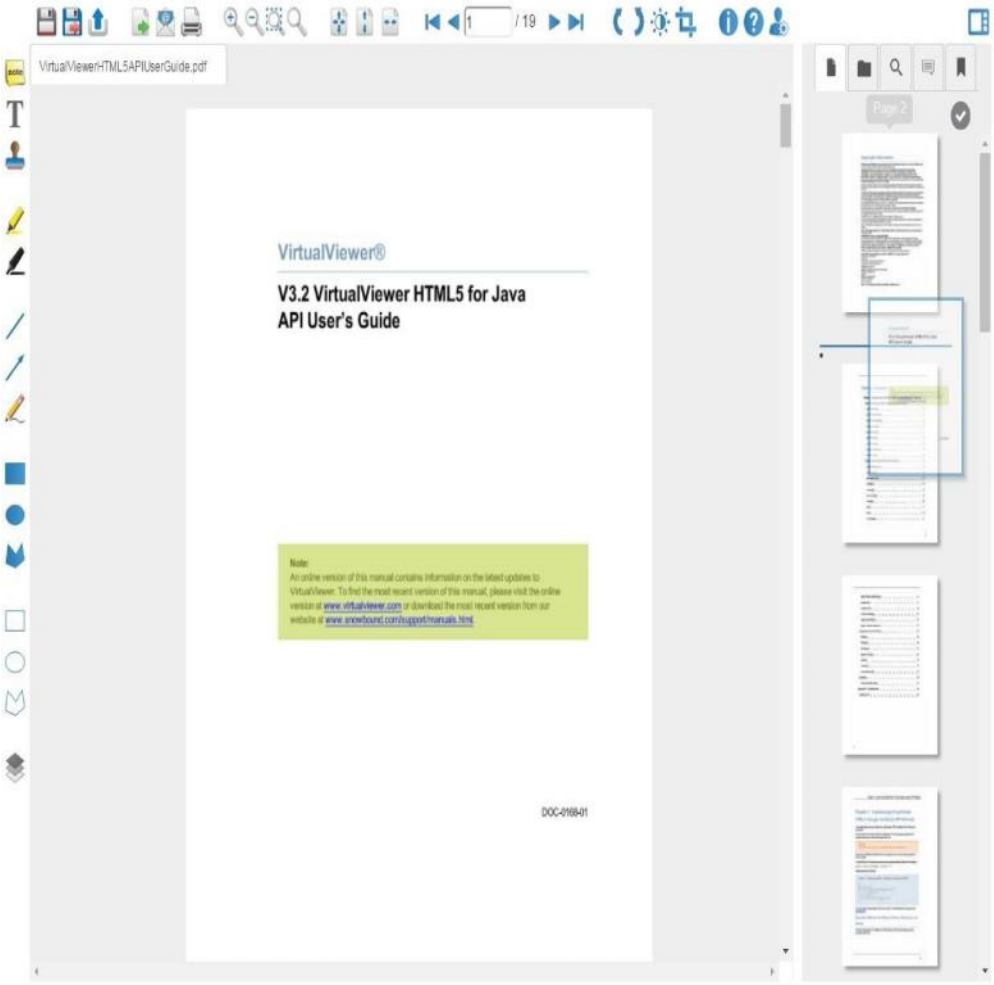
You can cut, copy, delete and insert a page from one document into another document open in the same instance of VirtualViewer HTML5 for .NET.

Dragging and Dropping Pages

Follow the steps below to use the drag and drop page manipulations feature:

1. Click and hold on the thumbnail that you wish to move and drag it up or down in the thumbnail panel.

2. A blue line appears horizontally in the thumbnail toolbar (in between thumbnails) indicating where the page being dragged will be placed in the document.
3. Let the mouse button go where you would like to place the thumbnail.
4. The page being dragged lands in between the two pages where the blue line was indicating the drop would happen.



To drag a thumbnail to an open tab, click and hold on the thumbnail that you wish to move and drag it to the open image tab. The page is appended to the last page.

Additional Notes

If the desired drop location is near the end of the document, drag the thumbnail to the bottom of the pane. The pane scrolls down as you reach the bottom.

You can select multiple pages with the Ctrl click and drag those in tan- dem. Dragging and dropping between sessions with two separate windows or browsers is now supported. Dragging and dropping between sessions functions the same way as the Copy feature.

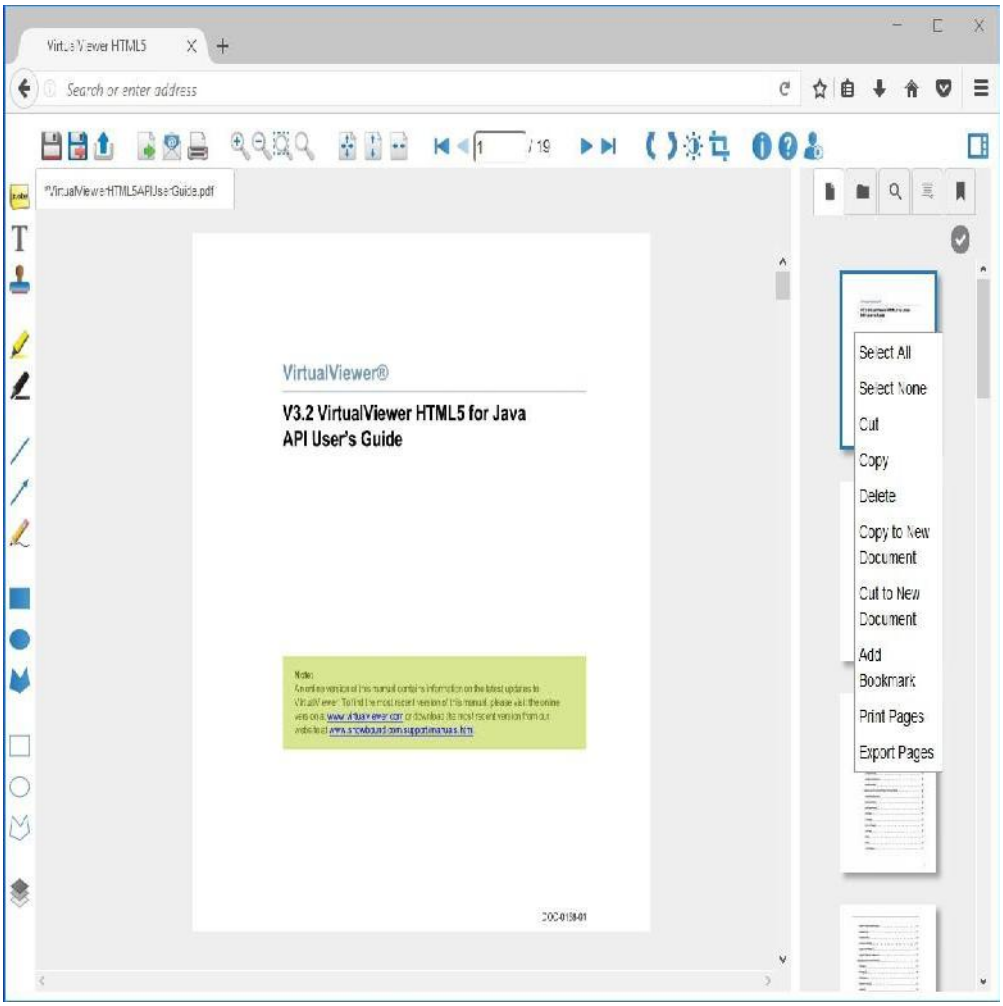
Saving Page Manipulations

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.

Rotate Specific Pages

Use the following APIs to rotate specific pages:

`virtualViewer.rotatePageBy(pageNumber, angle)` rotates the current page 0, 90, 180 or 270 (positive or negative) degrees from it's current state. So, you call this twice with 90 degrees as the parameter, the final image will be rotated by 180. It returns true if the page is rotated successfully. Otherwise, it throws an error.

`virtualViewer.rotatePageTo(pageNumber, angle)` rotates the document 0, 90, 180 or 270 degrees absolutely. Thus, if you call this twice with 90 degrees as the parameter, the final image will only be rotated by 90 degrees only. It returns true if the page is rotated successfully. Otherwise, it throws an error.

Page Manipulations Across Multiple Browser Sessions

You can now to perform Page Manipulations (Copy, Cut, Paste) across multiple browser sessions using local storage. Local storage is browser-specific, which requires the multiple sessions to be within the same browser.

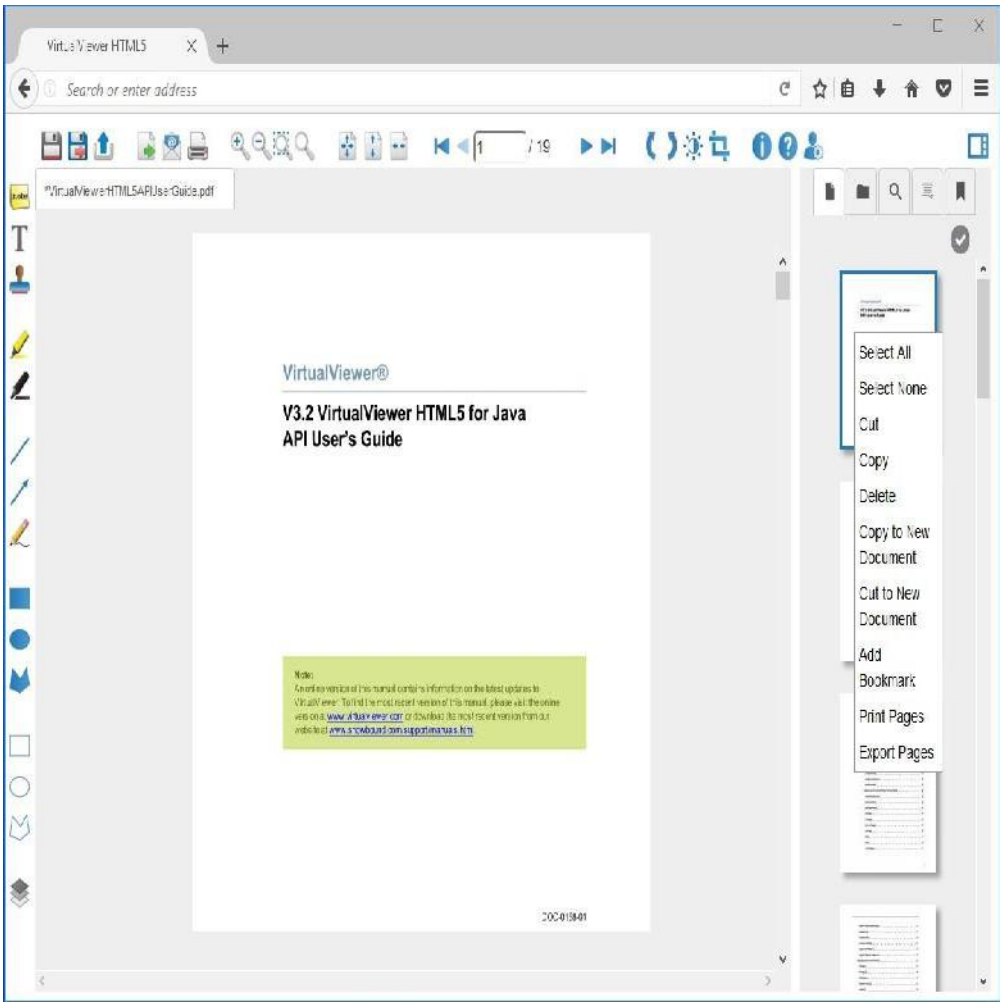
Saving Page Manipulations

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

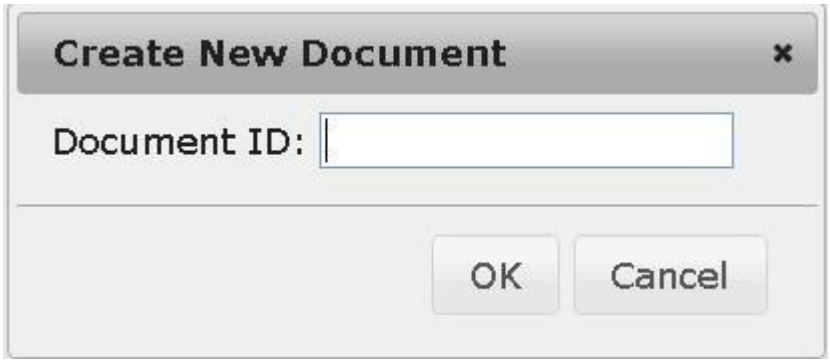
Copy to New Document

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.

Page Manipulations Across Multiple Browser Sessions

You can now to perform Page Manipulations (Copy, Cut, Paste) across multiple browser sessions using local storage. Local storage is browser-specific, which requires the multiple sessions to be within the same browser.

Customization

This section shows how to configure VirtualViewer® HTML5 on your system.

System Configuration

Configuring web.config

The web.config file contains a number of configurable tags. The **web.config** file is located in the **VirtualViewer** directory. It will only contain all of the .NET content server web.config parameters described in Server Tags for web.config. when the `contentServerType` parameter is set to integrated as shown in the following example:

Example 1.1: Set contentServerType Parameter to Integrated for web.config Parameter

```
<InitParams>
<add key="contentServerType" value="integrated"/>
</InitParams
```

Servlet Tags for web.config

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

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

Example 1.2: Set contentServerType Parameter to Integrated for web.config Parameters

```
Set contentServerType Parameter to Integrated for web.config Parameters
<InitParams>
<add key="contentServerType" value="integrated"/>
</InitParams>
```

This table lists and describes the AJAX Server web.config parameters.

Supported AJAX Server Parameters

Name	Default	Description
contentServerType	integrated	If set to integrated, the VirtualViewer HTML5 for .NET server will work as its own content server.
useXmlTran		If set to false, VirtualViewer HTML5 for .NET directly gets binary document data from the

This table lists and describes the ResponseServer parameters.

Table 3.1: ResponseServer

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

This table lists and describes the RequiredServlet servlet parameters.

Table 3.2: RequiredServlet Parameters

Name	Default	Description
tmpDir	`c:\tmp\`	Specifies a temporary directory for files created during the processing of page manipulation routines on the server.

This table lists and describes the optional servlet parameters.

Name	Default	Description
For AFP bit depth, please use modcaBitDepth.	N/A	To set the bit depth for AFP, please use modcaBitDepth. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
For AFP DPI, please use N/A		To set the DPI for AFP, please
For AFP format, please use	N/A	To set the format for AFP, please use
clientI		Sets the client image type.
contentHandler	`C:\Inetpub\wwwroot\ViewerNetContentServer\bin\FileContentHandler.dll`	Name of the content handler class to use.
default		Initial size of the byte array when saving to any format not
docxLicensePath	`C:\\user\\Support\\`	DEPRECATED - file path of the Office2010 license.

Name	Default	Description
	evaluation\\ Office2010.Total. Product.Family.lic	
do		The bit depth to use for Word documents. Valid values are 1 or 24. Must be set to 24 to display color output. Please also see wordBitDepth. Please note that increasing the bit depth is a tradeoff with performance. For more information on improving per- formance, please see Configuring to Maximize Your Performance or
docDPI	300	The DPI to use for Word documents. Must be set to 200 or more to display color output. Please also see wordDPI.
d o		The format to convert Word documents to. Valid values are TIFF_ G4, JPEG, TIFF_LZW, PNG. Please also see wordFormat.
doc- umentCacheAb- soluteExpiration	Infinite	Amount of time before a document is removed from the cache. This value must be either "Infinite" or a TimeSpan string format. The timeSpan method converts the string representation of a time interval to its TimeSpan equivalent. A simplified version of the format is { d | [d.]hh:mm[:ss] }, where dd = days, hh = hours, mm = minutes and ss seconds.
doc- umentC		Amount oftime after a

Name	Default	Description
ingExpiration		is last accessed before it is removed from the cache. If this value is set, doc- umentCacheAbsoluteExpiration must be unset or set to "Infinite." The value must be a TimeSpan string format the same as defined above.
fontMappingPath	`"C:/imgs/"/`	Sets MODCA file font mapping for AFP files.
ioc		The bit depth to use when decom- pressing IOCA pages. Valid val- ues are `1` or `24`. Please note that increasing the bit depth may neg- atively affect the performance. For more information on improving per- formance,
iocaDPI	300	The dots per inch (DPI) to use when decompressing IOCA pages.
io	TIFF	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 the client.
jp		Level of quality when a page is converted to JPEG and
logLevel	Finest	Detail of logging. Valid values: `Severe`, `Warning`, `Info`, `Config`, `Fine`, `Finer`, `Fin- est`, `All`

Name	e	Description
maxByte		Maximum number of times the byte array is doubled, if the ori- ginal estimate is too small,
maxRequestLength	4096	The attribute name in IIS under `httpRuntime`. When the user selects apppool running under .NET Framework 4, the `requestValidationMode` ="2.0" should be added in the con- figuration. For example: `<!-- httpRuntime maxRequestLength- h="512000" executionTimeout="7200" requestVal- idationMode="2.0"/ -->`
m		IIS uses `mimeType` to create the content type for the web doc- ument.This parameter is required in version 3.4 or greater. For example: `<mimeMap fileEx-`
modcaBitDepth	1	The bit depth to use when decom- pressing MO:DCA pages. Valid values are `1` or `24`. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Con- figuring to Maximize Your Per- formance or Quality.
m		The dots per inch (DPI) to use

		Description
		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`.
ov		Specifies the path of overlays used for AFP and MO:DCA files.
officeLicensePath	`c:\\inetpub\\VirtualViewer\\VirtualViewerNetHTML5\\Office2010.Words.lic, c:\\inetpub\\VirtualViewer\\VirtualViewerNetHTML5\\Office2010.Cells.lic, c:\\inetpub\\VirtualViewer\\VirtualViewerNetHTML5\\Office2010.Slides.lic`	Specifies the Office 2007 plug-in licenses for Word, Excel, and PowerPoint. **Note**: If you have multiple license files for Office 2007 for Word, Excel, and PowerPoint, you need to use the officeLicensePath parameter with a comma separated list as shown as the default value.
parsePath-		Passes the path of the Document Id from the server to the client
pclBitDepth	1	The bit depth to use when decompressing PCL pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
		The dots per inch (DPI) to use when decompressing PCL pages.
pclFormat	TIFF_G4_FAX	The format to convert PCL pages

to. Valid values are TIFF_G4_
FAX, JPEG, TIFF_LZW, PNG.

Name		Description
	e	The bit depth to use when decompressing PDF pages. Valid values are `1` or `24`. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
pdfDPI	200	The dots per inch (DPI) to use when decompressing PDF pages.
p		The format to convert PDF pages to. Valid values are `TIFF_G4_FAX, JPEG, TIFF_LZW, PNG`.
pixelLimit	N/A	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. Valid value formats `1000000` or **`1000x1000`**.
pp		The bit depth to use when decompressing PPT pages. Valid values are `1` or `24`. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
pptDPI	200	The dots per inch (DPI) to use when decompressing PPT pages.
p		The format to convert PPT pages to. Valid values are `TIFF_G4_FAX, JPEG, TIFF_LZW, PNG`.

Name	Default	Description

relativePaths	false	Allows relative paths to be used. For example:
		`<add key="useDllContentHandler" value="true" />`
		`<!--add key="contentHandler" value-e="C:/Inetpub/VirtualViewer/ Vir-tualView-erNetHTML5/bin/FileContentHandler.dll" /-->`
		`<add key="contentHandler" value-e="/bin/FileContentHandler.dll" />`

setSy		If set to true, VirtualViewer HTML5 for .NET automatically sets the viewer `\bin` path as the system path
`svgExclusions`	`RTF,PPTX,PCL_1,PCL_5, EMAIL`	Exclude any formats from SVG support. This is useful if a formats is misbehaving in SVG and you want to force that format to use the normal, bitmap delivery. The format is a comma-separated list of format names. The following are the valid values for the svgExclusions parameter: AFP, ASCII, DOT, DOTX, HTML, MO:DCA,PCL_1, PCL_5, PDF, PDF_15, POWER_ POINT, PPTX, ODS, ODT, RTF, XLS, XLSX, EMAIL Please note that the format names are case sensitive. The user can get the file type name for a given document by using the ImageInfo button and looking at the File Format value.
supportR		Turns on redaction

Name	Default	Description
thumbByteEstimate	6000	The initial byte size of the buffer used on the server to transport thumbnails.
tif		Initial size of the byte array when saving to TIFF to send to the client.
tiffCompressionType	`TIFF_G4_FAX`	Sets the accepted TIFF file formats. The values are: TIFF_LZW, TIFF_JPEG, TIFF_JPEG7, TIFF_G4_FAX. The defaultvalue is TIFF_LZW.
useDllContentHandle		If set to false, the content handler that you have pointed to will not be used.
wordBitDepth	24	The bit depth to use when decompressing Word pages. Valid values are `1` or `24`. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
		The dots per inch (DPI) to use when decompressing Word pages.
wordFormat	JPEG	The format to convert Word pages to. Valid values are `TIFF_G4_FAX, JPEG, TIFF_LZW, PNG`.
xl		The bit depth to use when decompressing XLS pages. Valid values are `1` or `24`. Please note that increasing the bit depth is atradeoff with performance.

Name	Default	Description
		For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
xlsDPI	200	The dots per inch (DPI) to use when decompressing XLS pages.
x		The format to convert XLS pages to. Valid values are `TIFF_G4_FAX, JPEG, TIFF_LZW, PNG`.
`Name`	`Default`	Description
For AFP bit depth, please use modcaBitDepth.		To set the bit depth for AFP, please use modcaBitDepth. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
For AFP DPI, please use modcaDPI.	N/A	To set the DPI for AFP, please use modcaDPI.
For AFP format, please use modcaFormat.	N/A	To set the format for AFP, please use modcaFormat.
clientImageType	N/A	Sets the client image type. Choose jpg or png.
conte	`C:\Inetpub\wwwroot\Vir-tual Vi`	Name of the content handler class to use.
defaultByteSize	40000	Initial size of the byte array when saving to any format not TIFF or

Name	Default	Description
		JPEG to send to the client.
docxLicensePath	`C:\\user\\Support\\evaluation\\\Office2010.`	**DEPRECATED** -file path of the Office2010 license.
docBitDepth	1	The bit depth to use for Word documents. Valid values are 1 or 24. Must be set to 24 to display color output. Please also see wordBitDepth. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.
		The DPI to use for Word documents. Must be set to 200 or higher to display color output. Please also see wordDPI.
docFormat	PNG	The format to convert Word documents to. Valid values are TIFF_G4, JPEG, TIFF_LZW, PNG. Please also see wordFormat.
fontMappingPath	`"C:/imgs/"/`	Sets MODCA file font mapping for AFP files.
iocaBitDepth	1	The bit depth to use when decompressing IOCA pages. Valid values are 1 or 24. Please note that increasing the bit depth may negatively affect the performance. For more information on improving performance, please see Configuring to Maximize Your Performance or Quality.

Table 3.3:

Table 3.4: RequestServer Parameters

Name	Default	Description
clearCacheOnSave	true	If set to true, clears the server document cache when a document is saved. When this parameter is set to true, the cache is cleared for the entire document when saving including annotations.
lo	"C:/Inetpub/VirtualViewer/VirtualViewerNetHTML5/virtualviewernetajaxser	Sets the client image type.
outputConfigPath	N/A	Specifies the path and name of the `output.properties` file which is used to determine the formats used when saving documents.
saveAnnotationsAsX		If true, saves annotations as XML rather than binary.
serverURL	"/VirtualViewerNetContentServer"	Sets the content server URL. This is ignored when `contentServerType` set to integrated.

Table 3.5: Retrieval Server Parameters

Name	Default	Description
preferencesPath	N/A	Specifies the location of stored client preferences on the server when using the default content

Name	Default	Description
		handler.

Table 3.6: MemoryCache Parameters

This section describes the parameters using VirtualViewer .NET's built-in MemoryCache which has a configuration section separate from our usual parameters in the web.config file.

Name	Default	Description
cacheMemoryLimitMegabytes	0	The maximum memory size, in megabytes, that an instance of a MemoryCache object can grow to. The default value is 0, which means that the MemoryCache class's autosize heuristics are used by default.
		Name of the cache configuration. For VirtualViewer.NET this should be set to **Default**.
physicalMemoryLimitPercentage	0	The percentage of physical memory that can be used by the cache. The default value is 0, which means that the MemoryCacheclass's autosize heuristics are used by default.
polli		A value that indicates the time interval after which the cache implementation compares the current memory load against the absolute and percentage-based memory limits that are set for the cache instance. The value is entered in "hh:mm:ss" format.

Customizing the User Interface

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

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

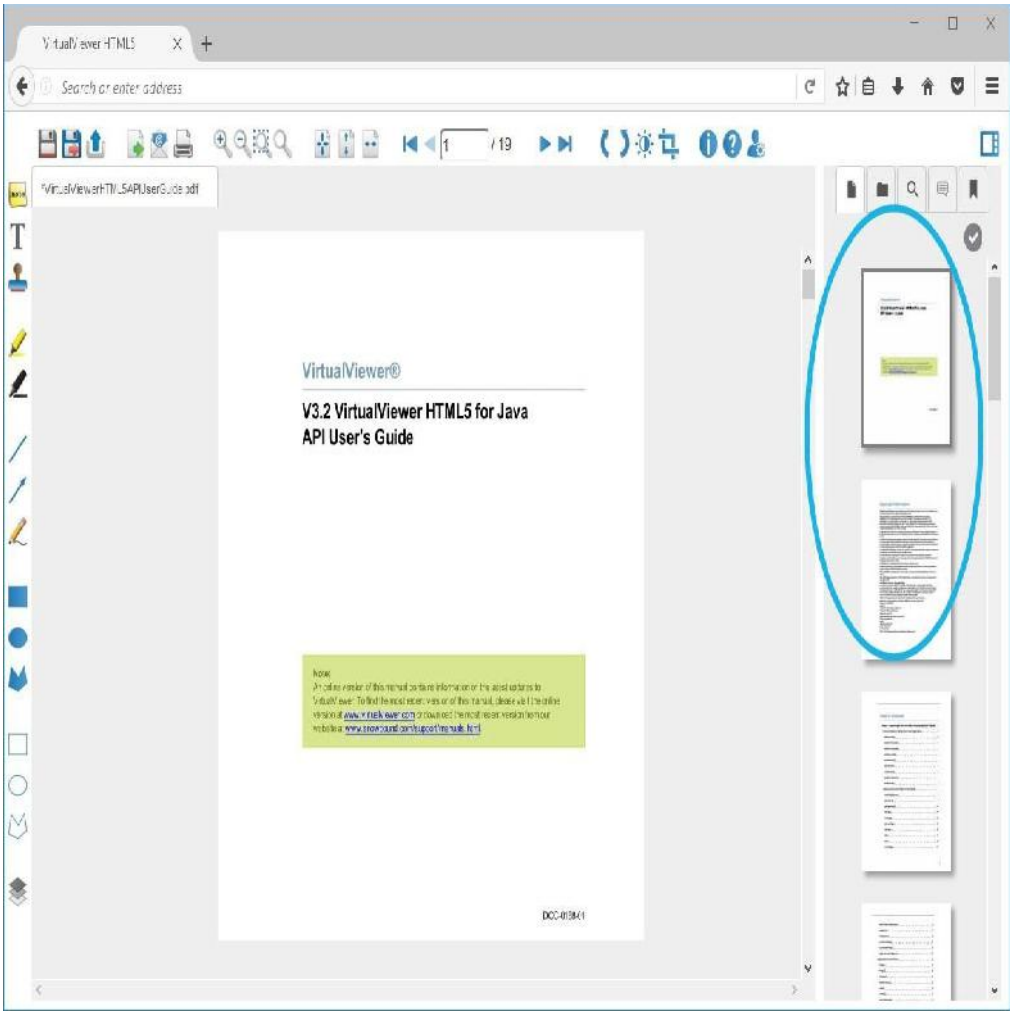
Example 1.3: Customizing What is Displayed inVirtualViewer HTML5

```
<!--
<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 Using VirtualViewer .HTML5.

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

Configuring the Pages and Document Panel Display



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

Please see 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 Connecting to Your Document Store .

The sample 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 1.4: Setting multipleDocMode to availableDocuments

```
This example shows how to set the multipleDocMode parameter in the con-
fig.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 getAvail-
ableDocumentIds() method description in Connecting to Your Document
Store
```

viewedDocuments

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

Example 1.5: Setting multipleDocMode to viewedDocuments

```
This example shows how to set the multipleDocMode parameter in the con-
fig.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:
&lt;body onload="myFlexSnap.initViaURL()"&gt;.
```

specifiedDocuments

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

Example 1.6: Setting multipleDocMode to specifiedDocuments

```
This example shows how to set the multipleDocMode parameter in the con-
fig.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 1.7: Changing multipleDocMode from availableDocuments to specifiedDocuments

```
This example shows how to change multipleDocMode from avail-
ableDocuments 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 spe-
cifiedDocuments and add a new line defining the array of specified doc-
uments:
var multipleDocMode = multipleDocModes.specifiedDocuments;
var SD = new Array("help.doc","info.tif","image.jpg);
In the index.html file, change the value of the onload event.
Results in index.html:
<body onload="myFlexSnap.initSpecifiedDocuments(SD);">
```

Config.js Parameters

You can use the config.js parameters to customize VirtualViewer HTML5 for .NET.

Descriptions of Config.js Parameters

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

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

Config.js Parameters

Config.js Parameters

Name	Default	Definition
servletPath	`"/VirtualViewer"`	Specifies the path to the servlet for the .NET configuration.
pictureControlsTimeout	200	Sets the wait in X milliseconds before requesting the adjusted images. This exists to throttle the number of image requests sent to the server. If the user quickly slides the sliders back and forth this will wait X seconds before requsting the updated image.
waitDialogTimeout	1000	Sets the wait in X milliseconds before displaying the "Please wait while your image is loaded." dialog message.
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.
invertedPanScrollX	true	Inverts pan scroll.

Name		Definitio n
invertedP		
con-tinuousScrollBufferSize	2	How many pages to buffer on each side of the document.
pageMani		Enable/Disable Page Manip-
ulationsNewDocu		Enable/Disable the "New Document" page
restric-tedPage Ma-		Array[String] of formats where page manipulations
enableTextExtraction	true	Determines whether to request text from the server
copySelectedText	true	Allows the copying of text.
guideLineWidth	2	Sets the guide line width.
activeGuideColor	"#000099"	Sets the active guide color.
lockedGuideColor	"#990000"	Sets the locked guide color.
s		If set to true, then screen can
screenSizes	[50, 50]	An array that sets the per-centages of each panel.
showAnnN		If set to true, shows the

		Definition
		If set to false, does not show the annotation
showAnnIndicators	false	If set to true, shows the annotation indicator. If set to false, does not show the annotation indicator.
sendDocu-mentWithAn		If set to true, includes annota-tions when
exportBurnAnnotations	false	Determines if VirtualViewer HTML5 should burn the annotations into the image when exporting.
oneLayerPerAnnotation		Creates a new annotation layer for each annota- tion. The ability to add a pre- fix and/or suffix to an annotation layer name using autoLayerPrefix and autoLayerSuffix String parameters when oneLayerPerAnnotation Boolean parameter is set to true in config.js. oneLayerPerAnnotation : true;
autoLayerPrefix	null	autoLayer Prefix: “snowPref Sets the annotation layer pre-fix.
autoL		Sets the annotation layer suf-
collapseStickiesSize	50	The default width in image pixels of the collapsed sticky

Name	Default	Definition
		note.
collapseStickies		Collapse stickies by default
immediatelyEditTextAnnotations	true	If set to true, newly added text annotations will immediately enter 'edit' mode with the contents highlighted. Sticky note and text box annotations will be immediately placed in edit mode once drawn on the screen so that the user can edit the text after being added to the page. If you do not want newly added text annotations to immediately enter 'edit' mode with the contents highlighted, set to false.
autoConfirmTextAn		f set to true, clicking on vvOuterDiv will auto-finish
`polygonNubSize`	`10`	Sets the color of the "handle" used to resize annotations and to indicate the "end zone."
`polygonNubSizeT`		Sets the color of the handle used to resize annotations and
polygonNubFillColor	`"rgba (0,0,255,.40)"`	Sets the color of the "handle" used to resize annotations and to indicate the "end zone."
base64Encode		Enables or disables Base64
rotateTextAnnotations	true	Determines if the text inside of text annotations rotate along with the document.

		Definition
enableTextRubberSt		When set to true, enables the Rubber Stamp functionality.
enableSingleClick ImageRubberStamp	true	When set to true, enables the single-click Image Rubber Stamp functionality
textRubb	{ textString: "Approved", fontFace: "Times New Roman", fontSize: 30, fontBold: true, fontItalic : true, fontColor: "00FF00" },	Configures the text rubber
stickyAnnButtons	{ "Sticky Note": false, "Rubber Stamp": false, "Bitmap": false, "Line": false, "Arrow": false, "Freehand": false, "HighlightRectangle": false, "FilledRectangle": false, "FilledEllipse": false, "FilledPolygon": false, "Rectangle": false, "Ellipse": false, "Polygon": false, "RedactionRect": false },	Allow the use of sticky Annotation Buttons. The Annotation will stay on until it is clicked again.

annotatio	//	The default appearance for a text annotation looks

	Default	Definition
	appearance options for annotations annotationDefaults: { fillColor: "FE0000", stickyFillColor: "FCEFA1", // yellowish stickyMargin: 10, // also need to adjust .vvStickyNote in webviewer.css highlightFillColor: "FCEFA1", highlightOpacity: 0.4, textString: "Text", fontFace: "Arial",	different look, the annotationDefaults config.js configuration parameter
enableAnnotationCommenting	true	Enable or disable the use of annotation commenting.
annot		List of strings that will be the options for If this list is empty, the user
defaultZoomMode	vvDefines.zoomModes.fitWindow	Sets the default zoom mode. You can use any of the following variables: fitWidth - Fits the page to the width of the image panel. fitHeight - Fits the page to the height of the image panel. fitPanel - Fits the page in the panel, regardless of

Name	Default	Definition
		landscape or portrait. `fitImage` - Fits the page to 100 percent.
fitLastBetweenDocum		If the user wants to retain the
zoomLevels	2,3,4,6,8,10,15, 20,30,40,50, 75,100,150,200, 300,400,600, 800,1000, 1500, 2000,3000	Defines the intervals at which to zoom.
defaultZ		Defines the default zoom level, if fitCustom is
maxZoomPercent	`1000`	Sets the percentage to stop allowing users to zoom the image.
zoo		Sets the wait in X milliseconds before requesting the zoomed image. This variable spares the server load by only requesting the
useBrowserScaling	false	If set to true, will use web browsers built-in scaling instead of more resource intensive (but higher quality) JS scaling.
serv		Force server only scaling. Set-ting this to true will increase performance on slow
singleThreadedSc-alingMode	"server"	For browsers which do not sup-port WebWorkers (IE9), which will degrade JS scaling per-

Name	Default	Definition
		formance. This setting will allow you to use for these browsers. Valid values are: "server" - use server-side scaling, will request new image at every zoom level. "js" - will use single-threaded JS scaling; not recommended.
defaultPrintingMethod	local	Sets which printing mechanism to use, local or server.
printBurnAnnotations	false	Determines if VirtualViewer HTML5 should burn the annotations into the image when printing.
enableDocumentNotes	true	Enable/disable the use of Document Notes
noteTemplates	`{ templateName: "Approve", templateString: "I approve this note" }, { templateName: "Reject", templateString: "I reject this note" }`	Note Templates
showThumbnailPanel		Sets the ability to hide the thumbnail panel and disable thumbnail
showPageThumbnails	true	Enables or disables the Pages thumbnail tab.
showDocTh		Enables or disables the Docu-
showSearch	true	Enables or disables the search tab.

Name			Definition
showBook-			Enable or disable the book- marks
searchCaseSensitive	false		Determines whether or not text searches should be case sensitive.
searchRedactionTags			Enable or disable tag results

		// Configure the keyboard
		key:'ctrl+shift+=,ctrl+shift+z,ctrl+-shift+plusKeypad', // ctrl+shift+= will not work in IE 9 or IE 10
		method:
		function() {
		virtualViewer.zoomIn();
		},
		localizedValue: 'hotkeyHints.zoomIn',
		defaultValue: 'Zoom In'
		},
		{
		key:'ctrl+shift+-,ctrl+shift+x,ctrl+-shift+minusKeypad', // ctrl+shift+- will not work in IE 9 or IE10
		method:
		function() {
		virtualViewer.zoomOut();
		},
		localizedValue: 'hotkeyHints.zoomOut',
		defaultValue: 'Zoom Out'
			Configures the keyboard short-

Name	Default	Definition
	},	
	{	

	key:'ctrl+shift+p',
	method: function() {
	virtualViewer.printDocument();	

	},	
	localizedValue: 'hotkeyHint- s.printDocument', defaultValue: 'Print Document'	
	},	
	{	

	key:'end',
	method: function() {
	virtualViewer.lastPage();	

	},	
	localizedValue: 'hotkeyHint- s.lastPage', defaultValue: 'Last Page'	
	},	
	{	

	key:'home',
	method: function() {
	virtualViewer.firstPage();	

	},	
	localizedValue: 'hotkeyHint- s.firstPage', defaultValue: 'First Page'	
	},	
	{	

	key:'ctrl+shift+pageup',
	method: function() {
	virtualViewer.previousPage();	

	},	
	localizedValue: 'hotkeyHint- s.previousPage', defaultValue: 'Previous Page'	
	},	
	{	

	key:'ctrl+shift+pagedown',
	method: function() {
	virtualViewer.nextPage();	

	},	
	localizedValue: 'hotkeyHint-	

Name	Default	Definition

	s.nextPage',
	defaultValue: 'Next Page'
	},
	{
	key:'ctrl+shift+l',
	method: function() {
	virtualViewer.rotateCounter();
	},
	localizedValue: 'hotkeyHint-
	s.rotateCounter',
	defaultValue: 'Rotate Left'
	},
	{
	key:'ctrl+shift+r',
	method: function() {
	virtualViewer.rotateClock();
	},
	localizedValue: 'hotkeyHint-
	s.rotateClock',
	defaultValue: 'Rotate Right'
	},
	{
	key:'ctrl+shift+t',
	method: function() {
	virtualViewer.toggleThumbnailPanel
	();
	},
	localizedValue: 'hotkeyHint-
	s.toggleThumbnailPanel',
	defaultValue: 'Toggle Thumbnail
	Panel'
	},
	{
	key:'ctrl+shift+c',
	method: function() {
	virtualViewer.copySelectedText();
	},
	localizedValue: 'hotkeyHint-
	s.copyText',
	defaultValue: 'Copy Selected Text'
	},
	{
	key:'ctrl+shift+d',
	method: function() {	

Name	Default	Definition

	vir-
	tualViewer.toggleColumnSelectionMode
	();
	
	},
	
	localizedValue: 'hotkeyHint-
	s.toggleTextSelectionMode',
	defaultValue: 'Toggle Column Text
	Selection'
	
	},
	
	{
	
	key:'ctrl+shift+u',
	method: function() {
	virtualViewer.toggleImageInfo();
	
	},
	
	localizedValue: 'hotkeyHint-
	s.toggleImageInfo',
	defaultValue: 'Toggle Image Info Dia-
	log'
	
	},
	
	{
	
	key:'ctrl+shift+g',
	method: function() {
	virtualViewer.collapseAllStickyNotes
	(true,true);
	
	},
	
	localizedValue: 'hotkeyHint-
	s.collapseStickyNotes',
	defaultValue: 'Collapse Sticky
	Notes'
	
	},
	
	{
	
	key:'ctrl+shift+n',
	method: function() {
	virtualViewer.collapseAllStickyNotes
	(false,true);
	
	},
	
	localizedValue: 'hotkeyHint-
	s.expandStickyNotes',
	defaultValue: 'Expand Sticky Notes'
	
	},
	
	{
	
	key:'ctrl+/',
	method: function() {
	virtualViewer.toggleKeyboardHints();
	
	},	

Name	Default	Definition
	localizedValue: 'hotkeyHint- s.showKeyboardHints', defaultValue: 'Show Keyboard Hints',	
maxInfoFi		Defines the field lengths that will bedisplayed in

imageInfoField	// Define the fields that will be displayed in the
	Image Info Dialog
	imageInfoFields : [// Define the *Document-Specific* properties here in the order they are to be dis-played
	{ fieldId: "documentId",
	fieldCaption: "Document ID"},
	{ fieldId: "documentDisplayName",
	fieldCaption: "Document Name" },
	{ fieldId: "documentByteSize",
	fieldCaption: "File Size (Bytes)"
	},
	{ fieldId: "pageCount",
	fieldCaption: "Page Count" },
	{ fieldId: "documentFormat",
	fieldCaption: "File Format" },
	
	// Define the *Page-Specific* prop-erties here in the order they are to be displayed
	{ fieldId: "compressionType",
	// This is only for Tiff files and will be 'TIFF_G4_FAX', 'TIFF_JPEG', 'TIFF_LZW', etc fieldCaption: "Com-pression Type" },
	{ fieldId: "imageSizePixels",
	fieldCaption: "Size in Pixels" },
	{ fieldId: "imageSizeInches",
	fieldCaption: "Image Size" },
	{ fieldId: "dpi", fieldCaption:
	"DPI" },
	{ fieldId: "bitDepth", fieldCap-tion: "Bit Depth" },
	{ fieldId: "pageNumber", fieldCap-tion: "Page Number" },
	{ fieldId: "tiffTag315", fieldCap-tion: "Copyright" }
	]
	};
		Defines the Page-Specific properties here in the order they are to be displayed.

reload Docum ent		Reloads the document model after a save. Used for sys-

		Definition
		umentId on
unsavedChangesNo-tification	true	Allows notification ofunsaved changes. A dialog box appears when closing a tab or browser.
enableEventNotificatio		Disables the sending of events to the server
multipleDocMode	vvDefines. multipleDocModes. viewedDocuments	Sets the multiple documents mode. You can use any of the following variables: availableDocuments- The getAvail-ableDocumentIds() is called in the contenthand-ler to populate the list of doc-uments. viewedDocuments - Docu-ments will be added to the set of documents as the user views them during the current VirtualViewer HTML5 session. specifiedDocuments- Uses an array of doc-umentIDs passed in as an array to myFlexSnap.initSpe-cifiedDocuments(). This is a replacement for initViaURL() in index.html. This is the default.
panI		Defines how many pixels to
enableSVGSupport	true	If set to true, the application will request SVG images from the server for supported formats. If set to false, the tra-

Name	Default	Definition
		ditional bitmaps will be requested.
enableSVGSupportF		If sset to true, enables SVG
enableConsolidateAnnotationLayer	true	Enables consolidation ofthe layers and creates the Master Layer.
enableCrop	true	Enables thecrop tool.
disableUploadDoc	false	Disables the uploaddocument button when set to true.
emai	`emailDefaults: { prepopulateFrom: "pre-populatedEmail@domain.com", prepopulateTo : "", prepopulateCC : "", prepopulateBCC : "", prepopulateSubject: "VirtualViewer Document attached",`	The default values for email
magnifierDefaults	`magnifierDefaults: { zoomPercent: 150, width: 300, height: 150, x: 200, y: 100, },`	The default values for magnifier functionality.
doNotLoadPageThum		If set to true, the viewer will never request any

		Def in
autoRes- izeTextAnnotations	false	If set to true, text annotations will automatically resize as you type.
	"help /help	Passed to `window.open` when creating the help win- dow. `window.open (helpURL,helpW in- dowName, helpWindowPara ms);`
helpWindowName	`"helpWindow";`	Passed to `window.open` when creating the help win- dow. `window.open (helpURL,helpWin- dowName, helpWindowParams);` This can be (and often should be) a relative URL Path.
helpWindo	`"scroll- bars=1,width=800,heig`	Passed to `window.open` when creating the help win- dow. `window.open (helpURL,helpW in- dowName, helpWindowPara ms);`

Hiding the Pages and Documents Panel

The Pages and Documents panel provides a convenient way to:

Navigate to any page in a document in the Pages panel.

Select another document to view from the multiple Documents panel.

Create a new document by dragging and dropping pages from another document.

However, this convenience does have a price. VirtualViewer performance degrades because it is processing every page in the document Pages panel and/or the first page of every document in the Documents panel. If you want to speed up performance, you may want to disable or hide the Pages and Documents 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 Text Edit Annotations

The Text 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 1.8: Configuring the Approved and Denied Rubber Stamps

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

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

- Font face: Arial
- Font size: 12
- Font color: #FF0000
- Font attributes: Normal

Configuring Image Rubber Stamp Annotations

The Image Rubber Stamp functionality is defined by the `customImageRubberStamps` parameter in the web.config file as shown below in a comma-separated list of names which will be used to pull the individual stamp configurations out of the web.config:

Example 1.9: Configuring Image Rubber Stamps

```
<init-param>
<param-name>testStamp1,testStamp2</param-name>
<param-value>8.5</param-value>
</init-param>
```

Set the comma-separated list of names which will be used to pull the individual stamp configurations out of the web.config. You use these names in the param-name tags as shown in this example:

Example 1.10: Configuring List of Image Rubber Stamps

```
<init-param>
<param-name>testStamp1</param-name>
<param-value>This is the First Test,300,175,ht-
tp://www.sample.com/sites/sample.com/files/images/Sample.png</param-
value>
</init-param>

<init-param>
<param-name>testStamp2<</param-name>
<param-value>This is the Second Test,600,300,ht-
tp://www.sample.com/sample.pngg</param-value>
</init-param>
```

The param-value tags are comma separated as follows:

displayName,**stampWidth**,**stampHeight**,**stampURL**

displayName - the text that will show up in the pop-up menu in the UI to describe the stamp.

stampWidth/stampHeight - the dimensions used when `enableSingleClickImageRubberStamp` is enabled in config.js.

stampURL - the URL to the stamp in question. This will be downloaded on servlet startup, converted to PNG, and stored in memory.

Set the `enableSingleClickImageRubberStamp` parameter to true in the config.js file to draw the bounding box when adding a rubber stamp to the image with a single click .It will be sized according to the dimensions specified in web.config. If false, it will behave like any other annotation.

Example 1.11: Configuring List of Image Rubber Stamps

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

Configuring the Magnifier

The Magnifier functionality is defined by the `magnifierDefaults` parameter in the config.js file as shown below with the default values:

Example 1.12: Configuring the Magnifier

```
magnifierDefaults: {
zoomPercent: 150,
width: 300,
height:150,
x:200,
y:100 },
```

The toggleMagnifier() method to close the Magnifier can be mapped to a shortcut key.

The API call virtualViewer.setMagnifierPosition(X, Y) overrides the defaults and will allow coordinates to be passed.

Configuring Default Annotation Values

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

Please see the following example:

Example 1.13: Configuring the Default Annotation Values

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

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


      highlightFillColor: "FCEFA1",
highlightOpacity: 0.4,

 textString: "Text",

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

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

- Line color
- Line width
- Fill color
- Sticky note fill color
- Sticky note margin
- Text string
- Font face
- Font size
- Font bold
- Font Italic
- Font strike

Configuring the Annotations Checkbox

To set 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;
```

Configuring Email Documents

To display the Email Document button, set the `emailDefaults` parameter to true in the config.js file as in the example below:

```
var emailDefaults = true;
```

Set the `prepopulatedForm` parameter to true in the config.js file as in the example below:

```
var prepopulateFrom = prepopulatedEmail@domain.com;
```

Set the parameters below in the **web.xml** file to set the values for your email system.

Example 1.14: Configuring Email

```
<init-param>
<param-name>smtpServer</param-name>
<param-value>...</param-value>
</init-param>

<init-param>
<param-name>smtpUsername</param-name>
<param-value>...</param-value>
</init-param>

<init-param>
<param-name>smtpPassword</param-name>
<param-value>...</param-value>
</init-param>
```

Print Dialog Box

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

```
var printBurnAnnotations = true;
```

Displaying the Include Annotations Checkbox

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

```
var printBurnAnnotations = true;
```

Displaying the Text and Non-text Checkbox

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

```
var printShowTypeToggles = true;
```

The default is to print both types.

Server Cache

VirtualViewer .NET has the ability to remove specific documents from the cache. VirtualViewer. NET uses the built-in MemoryCache, which has a configuration section separate from the server cache parameters in the web.-config file.

Added the following parameters allow memory control for .NET's ObjectCache standard in web.config:

enableDocumentCache - Determines whether document cache is used.

documentCacheAbsoluteExpiration - Amount of time before a document is removed from the cache. This value must be either "Infinite" or a TimeSpan string format. The timeSpan method converts the string representation of a time interval to its TimeSpan equivalent. A simplified version of the format is { d | [d.]hh:mm[:ss] }, where dd = days, hh = hours, mm = minutes and ss seconds.

documentCacheSlidingExpiration - Amount of time after a document is last accessed before it is removed from the cache. If this value is set, documentCacheAbsoluteExpiration must be unset or set to "Infinite." The value must be a TimeSpan string format the same as defined above.

Added the following parameters allow memory control for MemoryCache in web.config:

cacheMemoryLimitMegabytes - The maximum memory size, in megabytes, that an instance of a MemoryCache object can grow to. The default value is 0, which means that the MemoryCache class's autosize heuristics are used by default.

name - Name of the cache configuration. For VirtualViewer.NET this should be set to Default.

physicalMemoryLimitPercentage - The percentage of physical memory that can be used by the cache. The default value is 0, which means that the MemoryCache class's autosize heuristics are used by default.

pollingInterval - A value that indicates the time interval after which the cache implementation compares the current memory load against the absolute and percentage-based memory limits that are set for the cache instance. The value is entered in the hh:mm:ss format.

Additional Notes

This feature is not designed to solve all performance issues. It is designed to help the performance issues of cache-related calls such as Save, Save As, Send, Export, Email and Print. For example, it will help with saving large documents with large page counts (100+).

You can choose to limit the cache size by memory, but by default this feature limits the cache size by item count.

The **MemoryLimit** settings above are not hard limits. **MemoryCache** estimates memory usage and checks it against those limits every **pollingInterval** to decide whether to remove items from the cache.

This cache is dependent on the IIS application pool. If the application pool is recycled, which can happen if the website is not used frequently, all currently cached items will be cleared.

Displaying the Text and Non-text Checkbox

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

```
var printShowTypeToggles = true;
```

The default is to print both types.

Localization

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

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

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

Localization Files

Localized files are stored in the following directory:

```
../VirtualViewer/resources/locale/
```

The english file, named **vv-en.json**, located in that directory and can be used as a reference when translating to other languages.

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

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

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

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

Converting Terms

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

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

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

Supporting Accents/Special Characters

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

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

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

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

Example 1.15: Creating a French Language Translation File

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

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

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

```
"rotateLeft": {
"alt": "ROTATELEFT.ALT",
 "title": "ROTATELEFT.TITLE"
},
* In this case, we will use the same value for both the alt and title values.
```

The French translation forRotate LeftisRotation À Gauche.

```
Converting the accents/special characters in this translation into
Unicode results in:
Rotation &#192; Gauch
Using the converted results in vv-fr.json with:

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

Force a Specific Language

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

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

`..VirtualViewer/js/ vvDefines.js`

Example 1.16: Force a Specific Language

```
Remove the backslashes // before the word language and replace the val-
ues zz with the letter codes of the language file you want to force.
Have the translation file available for reference.
localizeOptions: {
//language: "zz",
pathPrefix: "resources/locale"
},
```

Using Keyboard Shortcuts

Certain VirtualViewer HTML5 for .NET functions can be accessed via keyboard shortcuts.

Select CTRL+/ to see a pop-up window with all of the keyboard shortcuts.

Please note the following about overriding and defining custom shortcuts:

The shortcut definition is based on JQuery.hotkeys.js.

Define more than one modifier (ALT, SHIF, CTRL) alphabetically For example: CTRL+SHIFT+u instead of SHIFT+CTRL+u.

For maximum browser compatibility and minimal shortcut conflict, Snowbound recommends using CTRL+SHIFT<char> in general when defining VirtualViewer HTML5 for .NET shortcuts.

The following table lists the available shortcut commands:

Table 3.7: Keyboard Shortcut Commands

Fu	Default Key	Parameter to Change the
Zoom In	CTRL+SHIFT+z	ZoomIn
Zoom Out	CTRL+SHIFT+x	ZoomOut
Invert	CTRL+SHIFT+i	invert
Flip Horizontally	CTRL+SHIFT+h	flipX
Flip Vertically	CTRL+SHIFT+V	flipY
Export Document	CTRL+SHIFT+e	exportDocument
Print Document	CTRL+SHIFT+p	printDocument
Last Page	end	lastPage
First Page	home	firstPage
Previous Page	CTRL+SHIFT+pageup	PreviousPage
Next Page	CTRL+SHIFT+pagedown	NextPage
Rotate Coun-	CTRL+S	rotate
Rotate Clockwise	CTRL+SHIFT+r	rotateClockwise
Show About Dialog	CTRL+SHIFT+a	showAboutDialog
Show Keyboard Shortcut Hints	CTRL+/	showKeyboardHints
Toggle Layer Man-	CTRL+SHIFT+m	toggleLayerM
Toggle Thumbnail Panel	CTRL+SHIFT+t	toggleThumbnailPanel
Fit Height	CTRL+SHIFT+j	fitHeight
Fit Width	CTRL+SHIFT+w	fitWidth
Fit Window	CTRL+SHIFT+q	fitWindow
Scroll Up By 33%	CTRL+pageup	scrollUpBy33Percent
Scroll Down By	CTRL+pagedown	scrollDownBy33Percent

Fu	Default Key	Parameter to Change the
Pan Left	left	panLeft
Pan Right	right	panRight
Pan Up	up	panUp
Pan Down	down	panDown
Thumb Page Down	CTRL+end	thumbPageDown
Thumb Page Up	CTRL+SHIFT+end	thumbPageUp
Toggle Horizontal Guide	CTRL+SHIFT+y	toggleHGuide
Toggle Vertical	CTRL+S	toggle
Toggle Crosshair Guide	CTRL+SHIFT+k	toggleCrosshairGuide
Copy Text	CTRL+SHIFT+c	copyText
Search Text	CTRL+SHIFT+f	searchText
Enter Pan Mode	CTRL+SHIFT+z	enterPanMode
Enter Select Text Mode	CTRL+SHIFT+insert	enterSelectTextMode
Enter Guide Mode	CTRL+SHIFT+home	enterGuideMode
Toggle Text Select Mode	CTRL+SHIFT+d	toggleTextSelectionMode
Toggle Image Info	CTRL+SHIFT+u	toggleImageInfo
Undo Annotation	CTRL+SHIFT+h	undoAnnotation
Redo Annotation	CTRL+SHIFT+y	redoAnnotation

Keyboard shortcuts are defined in config.js in the hotkeys section as in the example below:

Note:
When defining more than one modifier (alt, shift, ctrl), please specify them alphabetically For example: 'ctrl-shift-u' instead of 'shift-ctrl-u'. For maximum browser compatibility and minimal shortcut conflict, Snowbound recommends using 'ctrl-shift-<char>' in general when defining VirtualViewer shortcuts.

Example 1.17: Defining Keyboard Shortcuts

```
hotkeys: { zoomIn: 'ctrl++,ctrl+=',
          zoomOut:'ctrl+-,ctrl+_',
          exportDocument: 'ctrl+shift+e',
          printDocument: 'ctrl+shift+p',
          lastPage: 'end',
          firstPage: 'home',
          previousPage: 'ctrl+shift+pageup',
          nextPage: 'ctrl+shift+pagedown',
          rotateCounter: 'ctrl+shift+l',
          rotateClock: 'ctrl+shift+r',
          showKeyboardHints: 'ctrl+/',
          toggleThumbnailPanel: 'ctrl+shift+t',
          fitHeight: 'ctrl+shift+j',
          fitWidth: 'ctrl+shift+w',
          fitWindow: 'ctrl+shift+q',
          panLeft: 'left',
          panRight: 'right',
          panUp: 'up',
          panDown: 'down',
          thumbPageDown: 'ctrl+end',
          thumbPageUp: 'ctrl+shift+end',
          copyText: 'ctrl+shift+c',
          searchText: 'ctrl+shift+f',
          enterSelectTextMode: 'ctrl+shift+insert',
          toggleTextSelectionMode: 'ctrl+shift+d',
          toggleImageInfo: 'ctrl+shift+u'
},
```

Customizing VirtualViewer® HTML5 through JavaScript API Methods

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

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

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

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

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

Please see the example below:

Example 1.18: Customizing What is Displayed in VirtualViewer® HTML5

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

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

Advanced Customization

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

Virtual Documents

This section describes how to work with virtual documents.

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

Please note the following:

Exporting to a .tif may require significant resources especially if converting to 24-bit color.

If you are viewing all pages in a single document, you should not use Virtual Documents.

Document Notes is not supported in Virtual Documents.

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.

Virtual Document Syntax

The special identifier is the string `VirtualDocument:` which is then followed by any number of `documentIds`. The syntax can be used any time a

normal `documentId` could be used. A `documentId` in the comma-separated list may be specified in the following manner.

Table 4.1: Virtual Document Syntax

File Name	Description
ABC.tif	This specifies that all pages of the document should be included.
A	This specifies that only a single page from the doc-
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 1.1: Virtual Documents

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

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

Example 1.2: Virtual Documents

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

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

Virtual Documents: Save Document As

When a user prints, exports, emails, or uses Save Document As with a virtual document, the resulting document will reflect what the user sees on their screen at the time of execution. Please note that `sendDocument` is not supported in virtual documents. A work around is to send a virtual document with Save Document As. Save Document As has better functionality than `sendDocument`.

The loadVirtualDocumentAnnotations and saveVirtualDocumentAnnotations **web.xml** parameters enables virtual documents to read annotations from the source document and to save annotations created on the virtual documents back to the source document. The default values for both parameters are set to false

Printing Virtual Documents

To print a virtual document, select the Print button. 

Annotation Securing: 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.This allow each annotation layer to carry a set of permissions. These permissions allows the layer to be passed in with several different levels of permissions such as *read only* or *edit*.

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

Table 4.2: Permission Levels

Permission	Level	Actions Permitted
PERM_HIDDEN	Hidden	The layer is passed to the client but not displayed.
PERM_REDACTION*		This burns in the annotation layer
PERM_PRINT_ WATERMARK	Print Watermark	The user does not see the layer, but it will be burned in for printing.
PERM_ VIEW_ WATER MARK		View Watermark The user may view the layer, but may not hide the layer.
PERM_VIEW	View	The user may view or hide the layer.
PERM_PRINT	Print	The user may also print the layer.
PERM_CREATE	Create	The user may also add an object to the layer.
PER		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.

* Redaction annotations are only considered redactions when they are burned in and saved as an image format file such as TIFF format.

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 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.
	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,

Per	Definition
	delete an object, or rename or delete
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.
	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
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.

*Redaction annotations are only considered redactions when they are burned in and saved as an image format file such as TIFF format

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 1.3: Retrieving Annotation Layers

```
public Hashtable getAnnotationProperties (clientInstanceId, doc-
umentKey, annotationKey)
```

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

Key/Value Pairs

The **`permissionLevel`** will determine how the layer is handled. If no value is set, an exception will occur.

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

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

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

Saving Redaction Layers

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

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

Example 1.4: 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.

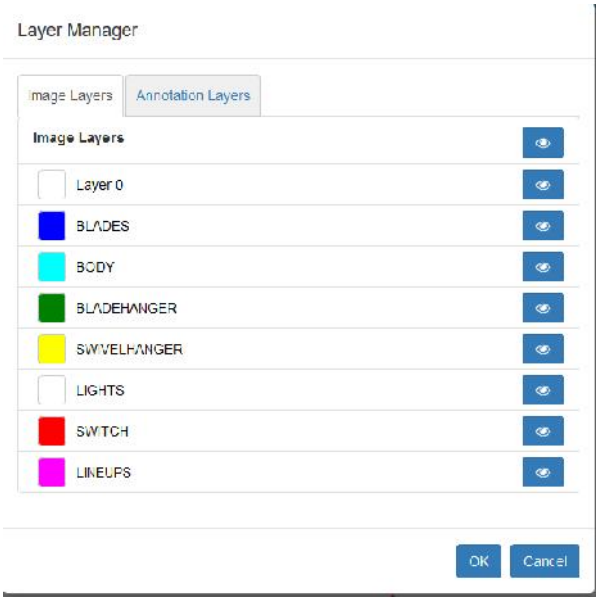
A layer which has been given a `permissionLevel` of `PERM_REDACTION` shall always print as part of the image, (since it has been burned into the image), even if the user chose to print without annotations.

DWG Layer Support

You can toggle DWG layers in and out of view in VirtualViewer. DWG layers are typically called referenced design layers. Layers include schematics or diagrams of blueprints that are embedded in the DWG file and laid over the image at view time.

Follow the steps below to use the **Layer Manager**:

1. Load a **DWG** file or a **blueprints** file that contains layers.
2. Select the **Layer Manager**  .
3. The Layer Manager dialog box displays a list of all the DWG layers on the **Image Layers** tab. The left side of the tab displays the layer name. The right side of the tab displays a visibility button to toggle the visibility of the single layer.
4. From the Layer Manager dialog box, choose which **DWG layers** to take in and out of view.



5. Select the **visibility button**  to view or hide the image layer. A check on the visibility button indicates that the image layer is hidden.
6. Select **OK** to display the changes that you made in the layer manager.

The DWG file format is only available for 64-bit Windows systems.

Cropping a DWG page with layers is not supported.

Selecting Export, Print, Save As, and Email includes all DWG layers. There is no option to choose specific layers for each function to carry out.

Page Manipulations carry over all DWG layers on that page.

Virtual Documents consisting of DWG pages allow you to take the layers in and out ofview on a page.

DWG xref Support

You can load a DWG file that contains references to other files (xrefs). Those related drawings are attached and displayed along with the DWG file. This feature assumes that the xrefs are in the same location as the original DWG file.

Set a valid directory in the `tmpDir` key in **web.config**.

If the content handler returns any external reference files, they are saved in the following directory: **[your temp directory]/[document ID]**. Make this directory accessible to VirtualViewer to read and write. External reference files are saved into these directories.

Content Handler

Use the following key in the content handler result class:

`KEY_EXTERNAL_REFERENCE_CONTENT_ELEMENTS`

This key returns a list of `ExternalReference` objects.

To implement external references in your content handler, include references to the `ExternalReference` class in your code.

Annotation Layers

The top level of the Annotation Layer Manager allows you to toggle all annotation visibility.

Follow the steps below to use the **Layer Manager** and add a new annotation layer:

1. Select the **Layer Manager**  . From the Layer Manager dialog box, choose the **Annotation Layers** tab.
2. Select the **Add** button to create a new layer.
3. In the active field, enter the **name** of the annotation layer.

4. Select the **Consolidate** button to consolidate annotation layers.
5. Select the **visibility button**  to view or hide the annotation layer. A check on the visibility button indicates that the image layer is hidden.
6. Click on the layer row to select the layer as active. When you draw annotations, they appear on the active layer.
7. If you want to edit the layer name, click on the **text of the layer name** or select the **edit pencil** button. Edit the layer name. Select the **OK** button to save the new name.
8. If you want to delete an annotation layer, select the **Delete** button.

Snowbound and FileNet Annotations

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

Configuring Snowbound Annotations

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

Example 1.5: Adding the annotationOutputFormat parameter Set to Snowbound

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

Snowbound Annotation Supported Configurations

Server

Table 4.3: Snowbound Annotation Supported Configurations - Server

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

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

Client

None

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

The menu-items for the layer will be visible but grayed-out in menus such as Select Layer.

When you right-click an annotation to edit it, the pop-up menu will simply not appear.

Configuring FileNet Annotations

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

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

Example 1.6: Adding the annotationOutputFormat Parameter Set to FileNet

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

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

Example 1.7: Setting oneLayerPerAnnotation to True

```
var oneLayerPerAnnotation = true;
```

FileNet Annotation Supported Configurations

Server

Table 4.4: FileNet Annotation Supported Configurations - Server

Parameter Name	Value	File Location
`annotationOutputFormat`	filenet	web.xml

Client

Table 4.5: FileNet Annotation Supported Configurations - Client

Parameter Name	Value	File Location
base64EncodeAnnotations	false	config.js
oneLayerPerAnnot		c

Configuring Daeja Annotations* (in development)

Please check for the latest status of this feature.

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

Example 1.8: Adding the annotationOutputFormat parameter Set to Daeja

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

Daeja Annotation Supported Configurations

Server

Table 4.6: Daeja Annotation Supported Configurations - Server

Parameter Name	Value	File Location
annotationOutputFormat	daeja	web.xml

Client

Table 4.7: Daeja Annotation Supported Configurations - Client

Parameter Name	Value	File Location
base64EncodeAnnotations	false	config.js

Annotation Mapping

The table below shows the FileNet annotation and its analogous Snow-bound Annotation

Table 4.8: 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

Watermark JSON Files

Watermarks for a document are stored in a json file. Like annotations, the file will be documentkey + suffix. For instance, 6-Pages-1.tif.watermarks.json. The .watermarks.json file is a list of json objects, so it has the format:
[{ myJsonData }, { myOtherJsonData }].

Each individual watermark is a json object. Each will have the following properties, formatted as seen in the attached example:

- *transparency*: A boolean. If true, the watermark will be transparent; if false, it will be a solid color.
- *adminCreated*: A boolean. If false, any user can manage any aspect of the watermark. If true, admin restrictions will apply (as described below).
- *text*: A string. This is the text that will appear on the watermark. Multiline watermarks are supported. This is done under the hood in the watermarks dialog, but if a user is manually entering json, they should enter a newline character ("\n") where a line break should be.
- *allPages:* A boolean. If this is set, the watermark will appear on every page of a document.
- *pages:* An array of page indices, zero-indexed. For instance, to place a watermark on only page one, this property would contain [0]. This is a key difference between watermarks and annotations. Watermarks are intended

to repeat across pages, so an identical watermark will have multiple pages it applies to.

- *widthAtTenPx:* An integer. This is a read-only value used by VirtualViewer to calculate the dimensions of the watermark, representing how wide the watermark is when the font is 10 pixels high.
- *stretch*: A double. This defines how far across the page the watermark will stretch. Set to 1.0, the watermark will go across 100% of the page (minus some margin space). Set to 0.5, 50% of the page. The UI allows only a small set of percentages. Diagonal watermarks will always stretch 100% across the diagonal.
- *format*: A json sub-object that has font and color information, as follows.
  -- font: A font name, for instance "Arial".
  -- color: We currently support only one color, so "000000" would be stored here.
- *position*: This is another sub-json object, that defines where the watermark will be placed on the page. There are two defining properties in here: the vertical placement of the watermark (top of the page, middle of the page, or the bottom of the page) and the direction of the text. While these options may open up further, the direction options are currently left-to-right text or diagonal text. The two options combine so that, for instance, top vertical placement & diagonal direction produce a watermark stretching from the top-left to bottom-right corner--while bottom vertical placement & diagonal direction will go from bottom-left to top-right.
  -- vertical: Use 0 for top, 1 for center, and 2 for bottom.
  -- direction: Use 0 for left-to-right text, and 2 for diagonal text.

Watermark.json file sample

[{"widthAtTenPx":19,"transparency":true,"adminCreated":false,"text":"bugs","allPages":true,"pages":[],"stretchPercent":0.5,"format":{"font":"Times New Roman","color":"000000"},"position":{"vertical":0,"direction":0}},{"widthAtTenPx":86,"transparency":true,"adminCreated":false,"text":"second%20watermark","allPages":false,"pages":[0],"stretchPercent":1,"format":{"font":"Times New Roman","color":"000000"},"position":{"vertical":2,"direction":2}},{"widthAtTenPx":62,"transparency":false,"adminCreated":false,"text":"sdadafsadfgsafd","allPages":false,"pages":[0],"stretchPercent":1,"format":{"font":"Times New Roman","color":"000000"},"position":{"vertical":2,"direction":0}}]

APPENDIX A: Tips

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

Documents Slow 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.`

Files and Thumbnails Slowly to Load

If it takes a very long time for files and thumbnails to load, please make sure that the **web.config** is pointing at the correct address and port for your web application.

Word, PDF and DWG Documents Do Not Display Correctly After Installation

If the viewer does not display DOC, PDF and DWG files as expected, please **restart IIS** and your **Windows system**. When the system Path environment variable has been modified to include the Snowbound installation directory specifying the location of the DOC, PDF and DWG plug-ins, the process that uses the Path variable needs to restart to pick up the new Path value.

DWG Documents Do Not Display Using URLContentHandler

If the viewer does not display DWG files as expected when you are using the URLContentHandler, please add DWG as an image type in your website's configuration. In IIS, select **MIME Types** and add **.dwg** as an **image/dwg** MIME type. Then restart IIS and try loading a DWG file.

Improving Performance or Quality

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

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

To improve the performance, you can save your files as black and white or grayscale. For example, if you are converting a PDF document, you can save the document in the TIFF_G4_FAX file format. This will make the file size smaller and improve performance. Please note that there is always a trade off between performance and quality. To improve performance, the quality of the image may be less. This is true whenever working with any imaging software. Please note that depending on the operating system and configuration, certain unusual or corrupt documents or files can cause the software to crash. Potentially, in some unusual circumstances, files may not be rendered identically to the creator application and may not format correctly or miss information.

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 Servlet Tags for web.config.

Example 1.1: Setting the Bit Depth

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

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

Table A.1: Bit Depth Parameter Values and Description

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

Setting the DPI - xxxDPI

This parameter determines how many DPI (dots per inch) should be used when converting a vector page. Typical settings for this parameter are 150, 200, or 300. The higher the DPI, the higher the quality of the image, but also the bigger the size, which means more processing on the server and larger page sizes across the network. The optimal setting for this varies by format,

but 200 is usually good for black & white documents or text, and 300 for color images and more detailed documents. Even higher numbers can be used (400, 600) but it can seriously affect speed of processing and available resources.

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

Example 1.2: Setting the DPI

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

The available DPI parameters are shown in the table below:

Table A.2: DPI Parameter Values and Description

Parameter Name	Description
docxDPI	The Dots Per Inch to use for Word 2007 documents.
io	The Dots Per Inch to use when decompressing IOCA
modcaDPI	The Dots Per Inch to use when decompressing MO:DCA pages.
p	The Dots Per Inch to use when decompressing PCL
pdfDPI	The Dots Per Inch to use when decompressing PDF pages.
p	The Dots Per Inch to use when decompressing PPT
wordDPI	The Dots Per Inch to use when decompressing Word pages.
x	The Dots Per Inch to use when decompressing XLS

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, see Servlet Tags for web.config.

Example 1.3: Setting the Format

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

The available format parameters are shown in the table below:

Table A.3: 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.
iocaF	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.
pclF	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.
pptF	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.

Parameter Name	Description
	The format to convert XLS pages to. Valid values are TIFF_G4_FAX, JPEG,
`xlsDPI`	The Dots Per Inch to use when decompressing XLS pages.

The full list of format server parameters and their usage is in Servlet Tags for web.config.

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 (or more) as shown in the following example:

Example 1.4: Displaying Color Output in Office 2007-2010

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

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

Default Configuration Maximizes Performance

Please note that the default configuration for VirtualViewer HTML5 for .NET 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 HTML5 for .NET, try setting the values of the following parameters in the `web.xml` file as shown below:

Example 1.5: 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.

Recommended JRE Memory Settings

Recommended JRE Memory Settings

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

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

- You go from black and white, to grayscale, to color documents (bits per pixel increases).
- You go from compressed to uncompressed document formats (lossy compression to raw image data).

You go from low resolution to high resolution documents (dots per inch / quality increases).

You go from small index card size images to large blueprint size images (number of pixels increases).

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

Decrease the number of documents you have open at any one time.

Decrease the quality of the images requested by decreasing bits per pixel, the resolution, or the size.

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

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

(height_in_pixels * width_in_pixels * (bits_per_pixel/ 8)) = image_size_in_bytes

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

Table A.4: 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	2550 * 3300 * (24 / 8) = 25245000

Image Size	Required Memory
300 dpi (2550 pixels by 3300 pixels)	bytes (25 megabytes)

Determining Memory Needed for the Number of Users and Pages Viewed in VirtualViewer® HTML5

To calcuate the amount of memory needed based on the number of users and potential pages viewed at any given time, use the example below:

The number of concurrent users * size per page in MB * 5 pages in view

For example, plug in the number of pages (in this case, 5) and the number of users (in this case, 1000):

black and white page (100 dpi) .1mb per page x 5 pages= .5 mb x 1000 users = 500 mb =~ 0.5 GB

black and white page (300 dpi) 1mb per page x 5 pages= 5 mb x 1000 users = 5000 mb =~ 5GB

color pages (300 dpi) 25mb per page x 5 pages = 125 mb x 1000 users = 125000 mb = ~122 GB

Capacity Planning

To make sure that VirtualViewer HTML5 for .NET performs well for your number of users, you may want to do some capacity planning.

Please see the following algorithm for calculating memory for 200 users:

For this calculation, we assume five pages will be active in VirtualViewer HTML5 for .NET at any one time - One page being displayed and four thumbnails.

> For 200 concurrent users viewing 5 1-bit color pages, the required Java heap capacity is 200 users * 5 pages * 1 mb /pg = 1,000 mB = 1GB.
>
> For 200 concurrent users viewing 5 24-bit color pages, the required Java heap capacity is 200 users * 5 pages * 24 mb /pg = 24,000 mB = 24GB.

For CPU size, generally, the larger the CPU and number of cores, the faster the response time. We expect VirtualViewer HTML5 for .NET to use all available resources to display the documents as quickly as possible. It will peg the CPU for short periods during heavy use.

For more information on the system requirements to run VirtualViewer HTML5 for .NET, please see System Requirements.

Caching to Improve Performance

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

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

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

Do You Need Caching at All?

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

Example 1.6: Turning Off Document Caching

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

Sizing the Cache if You Need It

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

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

Your page size in inches

Are the pages black & white or color?

Black & white uses 1-bit per pixel (bpp).

Color uses 24-bpp.

The desired resolution in dots per inch (DPI). 100 DPI is fine if the document will not be zoomed or printed. A higher DPI may be required if users are zooming in to look at details.

The size can be calculated using the following formula:

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

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

Here are some examples:

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

One black and white 8.5x11 inch page at 300 DPI = (8.5 inches* 300 dpi * 11 inches * 300 dpi) * (1 bit per pixel / 8 bits per byte) = 1051875 bytes or roughly 1 MB per page.

One color 8.5x11 inch page at 300dpi is 25245000 or roughly 25MB per page.

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

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

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

40 full 24-bit color pages at 300 DPI or

1,000 black and white pages at 300 DPI or

9,187 black and white pages at 100 DPI

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

Cache Maintenance

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

Example 1.7: Clearing Cache on Save

```
Clearing Cache on Save
<init-param>
<param-name>clearCacheOnSave</param-name>
<param-value>true</param-value>
</init-param>
```

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

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

Revalidate cache method called for every page

This is a short timespan cache to store answers from validateCache for each session/user. Every x minutes the cache will be deleted for each user (with storing and retrieval handled separately). This provides performance benefits to some users.

For whatever the specified window is (zero will check every time) we will cache the validation for that amount of time based on sessionId, documentId and HTTP action (GET or PUT). Once that time elapses, we will revalidate.

The time span value applies to both storage and retrieval.

There is a new initialization parameter "validationCacheExpirationMinutes" to control validation expiration. The default is five minutes.

When Does the Cache Get Cleared?

The cache gets cleared when the user clicks the Save button.

The parameter `clearCacheOnSave` in the web.xml must be set to true and the documentCacheSize in the web.xml must be greater than 0 to allow for caching.

To clear the Documentum caches if using Documentum, use the following Javascript method to clear the diskCache for the current session:

```
VirtualViewer.invalidateServerSessionDiskCache();
```

.Use the following Javascript method to clear the diskCache for all sessions:

```
VirtualViewer.invalidateAllServerDiskCaches();
```

Note:
This is assuming that `useSessionDiskCache` is not set to false in the web.xml.

When the Cache Size Is Reached

If you do not use the Save Document functionality, does the cache recycle on its own or does the cache get overwritten once the cache size is reached?

The cache size is determined by the `documentCacheSize` parameter set in web.xml. If the document is too big, it is not cached. The application will revert to using the `getDocumentContent` method in the content handler instead of retrieving the documents from the cache.

The `documentCacheSize` parameter limits the maximum size of a document (compressed) allowed to be put in the cache. The setting does not limit the overall cache size. The document size is logged if caching is on, log-level=finest and the document is too big to fit in the cache.

If the document is larger than `documentCacheSize` (in bytes), it will not enter the cache and VirtualViewer will log something such as:

```
For key: documentId=MyBig.TIF, data size(2710838) >
capacity (1024000). Not caching
```

When this happens, VirtualViewer will call `getDocumentContent` for the document every time it renders the main display or thumbnail since the document is not in cache. Calling `getDocumentContent` multiple times for

the same document is usually a significant and unnecessary performance hit which is why you should make the document cache big enough to hold your largest documents.

When the document is within the `documentCacheSize` limit, it is allowed in the cache. If the cache is already full, the cache will remove the oldest document(s) until there is enough room to put the new document into the cache. This is known as First In First Out (FIFO). The documents are identified by id, not by content. There is no pooling of documents with the same content and different ids.

The `documentCacheSize` parameter defines the maximum size of the heap allocated to VirtualViewer to use for document caching. If the -Xmx set in the web server for VirtualViewer is less, the cache will be limited to the smaller value. VirtualViewer may use any remaining heap memory for rendering pages and other operations.

Monitoring the Cache Size

Is there any way to monitor the cache size and have an alert when the cache size is about to be reached?

The system logs will say that the application cannot cache the document because the document exceeded the cache size. The application will retrieve the document with the content handler method **highe** `getDocumentContent` instead of retrieving from the cache.

Cache Setting in Tomcat

Tomcat has a setting to limit how much cache it can consume. Set this limit to stop the cache from growing at that threshold. Please keep in mind it does not clear the cache. It just limits it so there is no build up and eventual OOM exceptions.

In the Tomcat application server, find the **Tomcat Monitor** application. Open the Tomcat Monitor and then click on the **Java** tab. At the bottom, you will see the settings for Maximum memory pool.Use this to control not only how much is cached by one single application, but how much Tomcat will cache in total. You can therefore control how much VirtualViewer memory is cached as well as all other applications, thus managing memory at a much higher level.

Caching and Security

Snowbound Software has no mechanism to selectively remove cached content. However, a Cache Validation interface is provided so you can customize when cached content is permitted to be retrieved. You can implement the `validateCache` method to use your authentication system to validate

that the current user is authorized to access the cached page content. See the CacheValidator section in Connecting to Your Document Store.

APPENDIX B: Troubleshooting

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

Submitting a Support Issue

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

1. The configuration files: **index.html**, **config.js**, and **.\web.config**.
2. The document that the user is trying to view. Most issues are document specific.
3. 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 that is being used.

Troubleshooting with the vvCheck Diagnostic Tool

If you have any trouble installing or using VirtualViewer HTML5 for .NET, run the **vvCheck.exe** diagnostic tool included with your installation to troubleshoot your issue by checking that your system has the proper file permissions and settings to run VirtualViewer HTML5 for .NET. The vvCheck-.exe diagnostic tool is located in your main VirtualViewerNetHTML5 directory. To run vvCheck.exe, type the following in your command prompt:

```
vvcheck.exe http://-
localhost:8047/VirtualViewerNetHTML5
```

Please contact Snowbound Support at http://support.snowbound.com if you need any assistance using the vvCheck diagnostic tool.

"Please wait while your image is loaded" Message Dis- plays Indefinitely

In some cases, images do not load in VirtualViewer HTML5 for .NET, 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.ht-tpwatch.com) or Firebug (http://getfirebug.com). You can also use a stan-dalone application such as Fiddler (http://www.fiddler2.com) or Wireshark (http://www.wireshark.org) which can be run independently on the client machine. For Internet Explorer 9 users, the traffic can be captured using the IE Developer Toolbar (http://www.-microsoft.com/download/en/details.aspx?id=18359).

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

404 NotFound

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 con-fig.js contains the correct URL mapping to the AJAX servlet. If you changed the default directory name for VirtualViewer HTML5 for .NET 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 Descrip-tions of Config.js Parameters in AppendixE.

For VirtualViewer HTML5 for .NET, the web.config configuration should also be reviewed in addition to config.js. Make sure that the values for **<servlet-**

class> and **<url-pattern>** are correct for the relative **<servlet-name>**. Please note that by default, the servlet name is set to AjaxServlet.

405 Method Not Allowed

This error code indicates that the http request contains an action (e.g. POST, GET, HEAD, etc.) that is not allowed by the requested IIS server module. With respect to VirtualViewer HTML5 for .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-
="Sno-
owbound.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 HTML5 for .NET, 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.

Note:
If you are using the 64-bit release of VirtualViewer HTML5 for .NET, you must also disable 32-bit applications in your IIS Application Pool under **Advanced Settings > Enable 32-bit Applications**.

500 Internal Server Error

This error may occur if the content handler mapping is not correctly set in the web configuration. For VirtualViewer HTML5 for .NET for Java, check the `contentHandlerClass` parameter value. For VirtualViewer HTML5 for .NET, check the contentHandler key value. Make sure this value contains the correct path to the content handler. For more information on defining the `contentHandler` parameter, please see Optional Server Parameters.

Failed to Access IIS Metabase Error

If are having trouble getting a document to load and are getting a "Please wait while your image is loaded." message after going through the installation, follow the steps below:

This error typically occurs when you install IIS after you have asp.net installed.

To fix this issue, copy and paste the following into the Run Command (or command prompt) from the Start Menu:

```
%windir%\Microsoft.NET\Framework\vX.X\aspnet_
regiis.exe -i
```

If you do not have the file `aspnet_regiis.exe` at this location, you can find it on the Microsoft web site: http://msdn.microsoft.com.

Running this registration utility will reinstall `asp.net` for use with IIS and in most cases fixes the error stated above.

Snowbound DLLs Not Found at Runtime

If you are having trouble getting VirtualViewer HTML5 for .NET to work properly or display images properly, you may be able to resolve this issue by making sure that your Snowbound DLL files are found at runtime. This can be done in one of the following two ways:

1. Make sure the Snowbound directory containing the DLLs is included in the System PATH environment variable.

2. Copy the Snowbound DLLs to the /`Windows/System32` folder. Please note this work around has the drawback that the DLLs will not be removed by the Snowbound install and that future updates to the product may not work properly if they are not removed from the System32 directory before the update.

Note:
Please note that this is a temporary workaround for users where the application is having trouble finding the Snowbound Software format plugin files on their system. This workaround is only recommended until the issue can be resolved. For a production system, Snowbound Software does not recommend using this workaround to place files in the `/Windows/System32` folder

Annotations are Not Printed or Saved

If your annotations are not printed or saved, check that `printBurnAnnotations` and `exportBurnAnnotations` parameters are set to true in the config.js file. They may be set to false by default. If they are set to false, set them to true and refresh to resolve the issue.

Please see the following example for setting the parameters to true:

```
// Should we burn the annotations into the image when printing
var printBurnAnnotations = true;
// Should we burn the annotations into the image when exporting
var exportBurnAnnotations = true;
```

Images Disappear in Internet Explorer 9 When Zooming or Rotating

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

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

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

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

```
ie9DrawDelay: 900,
```

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

```
..Virtualviewer5/js/ vvDefines.js
```

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

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

Server Error: Failed to Execute URL

If you get the **Failed to Execute URL** server error in the `/VirtualViewerNetHTML5` Application when calling up VirtualViewer HTML5 for .NET from ISS, use the example below to disable VirtualViewerHandlerMapping.

In **web.config**, under .NET Framework 4, disable **VirtualViewerHandlerMapping** as shown in the example below:

Example 1.1: Disabling VirtualViewerHandlerMapping

```
<!--add name="VirtualViewerHandlerMapping" path="*" verb="*" mod-
ules="IsapiModule" scriptPro-
cessor="C:\Windows\Microsoft.NET\Framework64\v4.0.30319\aspnet_
isapi.dll" resourceType="Unspecified" requireAccess="Script"
preCondition="bitness64" /-->
<param-value>http</param-value>
```

You can also resolve this issue by removing the VirtualViewerHandlerMapping from the Handler Mappings in IIS when using the vvHTML5 .NET x64 4.0 Framework.

Using Integrated Pipeline Mode with VirtualViewer .NET x64

To use integrated pipeline mode with VirtualViewer® HTML5 for .NET x64, enable this line in **web.config** as shown in the example below:

Example 1.2: Enable Line in Web.config for Integrated Pipeline Mode

```
<add name="SnowboundAjaxServerHandler" path="AjaxServer" verb="*" type-
e="Snowbound.VirtualViewerNetAJAXServer.AjaxServerHandler" resourceType-
e="Unspecified" requireAccess="Execute" preCondition="integratedMode"
/>
```

Disable this line in **web.config** as shown in the example below:

Example 1.3: Disable Line in Web.config for Integrated Pipeline Mode

```
<add verb="*" path="AJAXServer" type-
e="Snowbound.VirtualViewerNetAJAXServer.AjaxServerHandler, Snow-
bound.VirtualViewerNetAJAXServer" />
<add name="VirtualViewerHandlerMapping" path="*" verb="*" mod-
ules="IsapiModule" scriptPro-
cessor="C:\Windows\Microsoft.NET\Framework64\v2.0.50727\aspnet_
isapi.dll" resourceType="Unspecified" requireAccess="None"
preCondition="classicMode,runtimeVersionv2.0,bitness64" />
```

URL Not Automatically Encoded in Internet Explorer 9

If you get a "no document found" error while accessing a document through Internet Explorer, verify that the URL does not contain any characters that are not allowed in a URL. You can use a link checker to validate and encode your URL, such as http://validator.w3.org/checklink

XLS or XLSX Page Content Truncated

Your XLS or XLSX page content may be truncated because XLS and HTML-formats do not include the page size in the document like Word and PDF. It can be set explicitly similar to how you can set the page size when printing. To set the page size to avoid truncated content, use the `xlsHeight`, `xlsWidth`, `xlsxHeight`, and `xlsxWidth` parameters in the **web.config** file as shown in the examples below. .

For XLS files set the parameters as shown in the example below to the height and width that you would like for your document:

```
<add key="xlsHeight" value="11" />
<add key="xlsWidth" value="14" />
```

For XLSX files set the parameters as shown in the example below to the height and width that you would like for your document:

```
<add key="xlsxHeight" value="11" />
<add key="xlsxWidth" value="14" />
```

APPENDIX C: Snowbound Error Codes

Detailed Status/Error Codes

This table lists the possible Snowbound errors and their descriptions.

Error Codes

	Error Code	Description
OUT_OF_MEMORY	-1	Failed on memory allocation. Problem with a standard memory allocation. For more information on determining memory requirments, please see Determining Memory Requirements.
FILE_NOT_F		Open call failed when trying to decom-
CORRUPTED_FILE	-3	File format bad, or unreadable.
BAD_S		String passed in is null or
BAD_RETURN	-5	Internal DLL problem. Submit a support issue at http://support.snowbound.com and attach the document you were pro- cessing when you received this error.
CANT_CREATE		Fail on saving when attempting to cre- ate a new file. On this error check that you have permission to write to that dir- ectory and
FORMAT_NOT_ ALLOWED	-7	Image was not recognized as a format the library can decompress.
NO_BITMAP_		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,
DISK_FULL	-9	Error writing data to the disk. Standard

Error Code		Description
		file i/o write failed.
BAD_DISPLAY		Tried to display with negative coordin-
PAGE_NOT_FOUND	-11	Used for multi-page file format support when attempting to access a page which does not exist. This error code provides information of an empty Word-page which is not converted to an empty page in PDF or TIFF.
DISK_READ_E		File format was truncated and tried to read past end of file. Standard
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_OPE NING_		Bad scanner driver or driver not con-
CANT_FIND_TWAIN_DLL	-17	TWAIN scanner driver not installed or not found (TWAIN.DLL).
USER_C		Cancel out of low level save or low level decompress. Usually not an error but termination of a
EVAL_TIMEOUT	-19	Date on an evaluation copy of the Snowbound product has expired.
USING_RU		Version not allowed for design
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 pro-cessing function on an image whose bits per pixel is not allowed. Please see Appendix D, Supported File Formats for the pixel depths of each supported format.

	Error Code	Description
PALETTE_IMAGES_	-22	Some image processing operations
NO_LZW_VERSION	-23	No LZW or GIF code in this version.
DLL_NOT_L		DLL not loaded for Win 3.x version. There was an error loading a DLL. Please open a support issue at http://support.snowbound.com
FORMAT_WILL_NOT_OTFLY	-25	Format will not support on the fly decompression.
NO_TCOLOR_F		No transparency color information
COMPRESSION_NOT_SUPPORTED	-27	Currently not supporting this compression format.
NO_MORE_		Returned when scanning has completed all pages in the
FEEDER_NOT_READY	-29	No more pages ready in document feeder.
NO_DELAY_TIME_		No delay time was found for the anim-
TIFF_TAG_NOT_FOUND	-31	Could not find the .TIF tag.
NOT_A_TILED_		Not recognized as a TIFF tiled
NOT_SUPPORTED_IN_THIS_VERSION	-33	You are using a version that does not support this function. You do not have support for this file format. You may contact support or your account representative to get information on the RasterMaster option that will allow you to process the file format.
AUTOFEED_		Autofeed fail in the TWAIN
NO_FAST_TWAIN_SUPPORTED	-35	TWAIN driver cannot do fast transfer.
NO_PDF_VE		The PDF processing option was not

	Error Code	Description
		option, please make sure the name of the directory containing Snowbound's pdfplug.dll is in
NO_ABIC_VERSION	-37	No ABIC plug-in code in this version.
EXCEPTION_E		Internal error. An exception occurred during processing. Please enter a sup- port ticket at http://sup-port.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
NO_VECTOR_ CAPABILITY	-39	No vector plug-in found in this version.
NO_PCL_VE		The PCL processing option was not found. If you have the PCL processing option, please make sure the name of the directory containing
NO_JPEG2000_VERSION	-41	NO JPEG2000 plug-in found in this version.
SEARCH_STRING_NOT_	-42	Did not find attempted search
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_		This file was password
METHOD_NOT_FOUND	-45	The Snowbound method was not found. Please check the spelling of the

	Error Code	Description
		method name and Snowbound library version.
ACCESS_D		Access denied. Please check the secur-
BAD_LICENSE_PRIMARY BAD_LICENSE_SECONDARY	-47	Primary level license loaded is bad. Secondary level license loaded is bad.
PASSWORD_		This file was password protected for
OOXML_LICENSE_NOT_FOUND	-52	The OOXML license file was notfound.
OOXML_LICENSE_		The OOXMLlicense file expired or is

General Error Define Values Retrieved from Status Prop- erty

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

Table D.1: General Error Define Values Retrieved from Status Property

		Desc
GENERAL_STATUS.SYSTEM_CRASH	-100	If an internal exception is thrown, this is the resulting value.
GENERAL_STATUS.		Image data of the
GENERAL_STATUS.DEFAULT	-102	What the internal values are initially set to

		Desc
GENERAL STATUS.		Operation completed
GENERAL STATUS. SNOWBND_ERROR	-1	Operation failed. See `StatusDetails` property.
GENERAL_STATUS. IMAGE_NOT_AVAILABLE	-103	Internal image data unavailable when trying to complete
GENERAL STATUS. SNOWBND_API_NOT_ AVAILABLE	-104	API is not implemented
GENERAL STATUS.		Parameter is
GENERAL STATUS. DISPLAY_ERROR	-106	General error display

General Status/Error Codes

Error	Description
DELETE_ERROR	The image in memory cannot be removed.
DISPLAY_E	Any problems with displaying an
IMAGE_NOT_AVAILABLE	No image data is available to do manipulations on.
NOT	This is returned if a parameter
SNOWBND_API_NOT_AVAILABLE	This is returned if an API method is not implemented in the current build.
SNOWBND_E	General API error code of an unsuc-
SNOWBND_OK	General API status of a successful action.
SYSTEM_	This is returned when a Critical

APPENDIX D: Supported File Formats

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

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

Table C.1: File Format Key

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

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

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

If you have any questions about what format to select you may contact Snowbound Technical support on the web at http://support.snowbound.com. We do our best to support product and document specifications and to work in

common platform environments, however there are always exceptions. If you find an exception please contact Snowbound Support to let us know about it.

Table C.2: Supported File Format Descriptions

B = Base

D = In Development

O = Optional format. Additional charges may apply.

		File Type Number	Input Bit	Common Bit	Win Read	Win Write	Supports Text Search
ABIC (reading)	IBM image compression for scanned checks. Note: Not yet supported with RasterMaster .NET x64 or RasterMaster DLL x64.	46	1		O		No
AFP (MO:DCA)	Advanced Function Presentation™. IBM format which uses CCITT G3, G4, and IBM MMR formats. 1-bit only. This is a multi-page file format	74	1, 24	1, 24	.afp	O	O	Yes
ASCII	Snowbound reads in ASCII text files and converts them to a bitmap. You may get a -7 FORMAT_NOT_ ALLOWED error when trying to convert the ASCII text format.	38	1	No	.txt	B		Yes
BMP_ COMPRESSED	Originated by Microsoft, BMP supports 1, 4, 8, and 24-bit images.	12	4, 8	4, 8	.bmp	B		No
BMP_ UNCOMPRESSED	Originated by Microsoft, BMP supports 1, 4, 8, and 24-bit images.	1	1, 4, 8, 16, 24	1, 4, 8, 16, 24	.bmp	B	B	No
BROOK_TROUT	Brooktrout FAX format.	29	1		.brk	B	B	No
CALS	Government specified format.	18	1		.cal	B	B	No
CCITT_G3	Group 3 compression for bitonal (1-bit) image data.	33	1		.tif	B	B	No
CCITT_G3_FO	Group 3 compression for bitonal (1-bit) image data.	53	1		.tif	B	B	No
CCITT_G4	Group 4 compression for bitonal (1-bit) image data.	34	1		.tif	B	B	No
CCITT_G4_FO	Group 4 compression for bitonal (1-bit) image data.	52	1		.tif	B	B	No
	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	83	1 , 8, 24	1 , 8, 24	.cif	O		No

		File	Input Type	Output Bit		Win	Win	Supports Text Read Write
	operators with multiple arguments, various pre-defined default values, more efficient allotment of encoding values and shared subroutines within a FontSet (family of fonts).							
CIFF	Camera Image File Format is a raw image format designed by Canon.	81	1 , 8, 24	1 , 8, 24	.cif	O	-	No
CIMS (ABIC)	Check Image Management System. Developed by Carreker. Same as ABIC. Note: Not yet supported with RasterMaster .NET x64 or RasterMaster DLL x64.	80	1	No		O		No
CLIP	Microsoft Windows clipboard format.	27	1, 4, 8, 24	1, 4, 8, 24, 32	.cip	B	B	No
COD	Liberty IMS black and white format.	72	1	No	.cod	B	B	No
CSV	Comma separated value list, a text spreadsheet.	99	1,24	No	.csv	O		No
CUT	Cut images are only 8 bits per pixel and the palette is stored in a separate file. Originated by Media Cybernetics.	31	8	No	.cut	B		No
DCS	The DCS format is a standard Quark Express Format. Each plane is stored as an EPS record.	62	32	32	.dcs	B	B	No
DCX	Intel created this format as a multi-page .PCX format. Each page is a .PCX file in whole which can be 1, 4, 8, and 24-bit.	11	1, 4, 8, 24	1, 4, 8, 24	.dcx	B	B	No
DIB	Standard Windows Device Independent Bitmap. Supports 1, 4, 8 and 24-bits. This is a multi-page file format.	48	1, 4, 8, 24	1, 4, 8, 16, 24, 32	.dib	B	B	No
DICOM	Medical image format supporting 1, 12, 16, and 24 pixel images. Limited support.	55	8, 16, 24	No	.dcm	B		No
DOC	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	86	1, 8, 24, 32	No	.doc	O		Yes

		File Type Number	Input Bit	Common Bit	Win Read Ext.	Win Write	Supports Text Search
	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.						
	DOCM is an MS Open Office XML				.docm	O	
DOCX	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.	93 1, 8, 24, 32	No	.docx	O	-	Yes
	Autodesk® AutoCAD® format. Used for computer aided design (CAD)	90	0				

						Win	Win	Supports Text Read Write
	The DWG format can be read in the VirtualViewer .NET Content Server.							
DXF	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 The DWG format can be read in the VirtualViewer .NET Content Server.	91	0	24	.dxf	O		No
EMAIL	E-mail message created with MS Outlook.	89	1		.msg	O	-	No
EPS (preview)	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.	14	1, 4, 8, 24	1,8, 24, 32	.eps	B	B	No
EPS_BITMAP	EPS Compressed bitmap format. It is an Adobe encapsulated Postscript file with either G4 or JPEG data embedded.	63	8, 24, 32	1, 8, 24, 32	.eps	B	B	No
EPS_BITMAP_G4	EPS Compressed bitmap format. It is an Adobe encapsulated Postscript file with either G4 or JPEG data embedded.	64	No	1, 8, 24, 32	.eps		B	No
EPS_BITMAP_LZW	EPS Compressed bitmap format. It is an Adobe encapsulated Postscript file with either G4 or JPEG data embedded.	69	No	1, 8, 24, 32	.eps		B	No
FileNet	Image format developed by FileNET Corporation for viewing documents.	78	1			B		No
FLASHPIX	24-bit tiled JPEG format that includes multiple resolution images.	54	8, 24	No	.fpx	B	-	No
GIF	Created by CompuServe for compressing 2, 3, 4, 5, 6, 7, and 8-bit palette images. Uses the LZW algorithm.	4	2, 3, 4, 5, 6, 7, 8	4, 8	.gif	B	B	No
GIF_INTERLACED	Same as GIF		44	1,				B

		File	Input Type Bit Num		File Ext.	Common Win Read	Common Win Write	Supports Text Search
	the raster data in an interlaced							
GX2	Originated by Brightbill Roberts for ShowPartner DOS applications. Supports 4 and 8-bit images. Simple run length encoding technique.	22	4, 8	No	.gx2	B	-	No
	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	82						-
ICONTYPE	Microsoft icon format. Contains a standard device independent bitmap. Supports 1 and 4 bits uncompressed.	25	1, 4	No	.ico	B		No
IFF_ILBM	Used on the Commodore Amiga computers for native bitmap format. Uses a run length format for 1, 4, and 8-bit palette images.	26	1, 4, 8, 24	1, 4, 8, 24	.iff	B	B	No
IMG	Originated by Digital Research for storing 1-bit images.	28	1	No	.img	B		No
IMNET	IMNET G4 compressed format.	42	1	No		B		No
IOCA (MO:DCA) *	Image Object Content Architecture. IBM format which uses CCITT G3, G4, and IBM MMR formats. 1-bit only. This is a multi-page file format.	24	1, 24	1	.ico	O	O	No
JBIG	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.	71	1	1 (with plugin)	.jbg	B		No
JBIG2	JBIG2 is a highly-compressed black and white image format that uses symbol recognition and substitution for very dramatic compression results. Snowbound's viewers and conversion programs can be used to directly view JBIG2 documents or	77	1	1 (with plugin)	.jbg	B	B	No

			Number			Win	Win	Supports Text Write Search
	convert those documents to a variety of output formats.							
JEDMICS	US Military CCITT G4 tiled image format for storing Government documents and drawings. Supports 1-bit per pixel.	56	1		.jed	B	B	No
JPEG	Joint Photographics Experts Group. This was a group spearheaded by Kodak for 24, 32, and 8-bit gray scale lossy compression. Lossless JPEG not supported.	13	24, 32	8, 24, 32	.jpg	B	B	No
JPEG2000	JPEG2000 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.	70	8, 24	8, 24	.jpg2	O	O	No
KOFAX	Kofax Format.	23	1	No		B		No
LASER_DATA	Compression for documents originated by LaserData Corp. 1-bit images only.	19	1	No		B		No
LINE_DATA	Presents data for each variable on a single line.	75	1			B		No
MACPAINT	Original Apple bitmap file format. All MacPaint images are 720 x 576 pixels 1 bit.	21	1	No				No
MAG	Mag Format.	61	1	No	.mag	B		No
MO:DCA	Mixed Object: Document Content Architecture . IBM format which uses CCITT G3, G4, and IBM MMR formats. 1-bit only. This is a multi-page file format.	49	1, 24		.mca	O	O	Yes
MSP	Microsoft Paint program bitmap file format. Supports 1-bit images . Uses a type of RLE compression found also in compressed .BMP files.	30	1	No	.msp	B		No
NCR	A simple header with CCITT group 4 data.	65	1	No	.ncr	B		No
ODF	Open Document Format is an XML-based file format for representing electronic documents such as spreadsheets, charts,	98	No	No	.odf	O		No

		File Type Number	Input Bit	Common Bit	Win Read Ext.	Win Write	Supports Text Search
	presentations and word processing documents.						
ODP	Open Document Format for presentations.	101	No	No	.odp	O	-	Yes
ODS	Open Document Format for spreadsheets.	97	1, 24	No	.ods	O	-	Yes
ODT	Open Document Format for word processing (text) documents.	96	1, 24	No	.odt	O	-	Yes
OOXML	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.	94	1, 8, 24	No	-	O	-	Yes
PCL_1	Hewlett Packard printer file format. Support for color and grayscale output. Supported as a separate plugin. This is a multi-page file format.	57			.pcl	O	B	
PCL_1 (without plugin)	Hewlett Packard printer file format. Support for color and grayscale output. Supported as a separate plugin. This is a multi-page file format.	57	No	1	.pcl	O	B	Yes
	Hewlett Packard printer file format. Support for color and grayscale output. This is a multi-page file format.	76	No		.pcl	O	B	
PCX	Zsoft bitmap file format. Similar to pack bits compression. Supports 1, 4, 8, and 24-bit images.	2	1, 4, 8, 24	1, 4, 8, 24	.pcx	B	B	No
PDF	Portable Document Format. File format developed by Adobe to capture formatting information from a variety of desktop publishing applications.			1, 2, 1, 2 4 4,	.pdf	O	B	

		File	Input	Output		Win	Win	Supports
			Type	Bit				Text Read Write
	send formatted documents and have them appear on the recipient's monitor or printer as they were intended. **Compatible with the PDF/A-1b – Level B (basic) conformance specification and conforms to PDF v1.7**. JPEG2000 objects within a PDF file require Snowbound Software's optional JPEG2000 license. Supports some types of Adobe specified PDF annotations, however does not support XFA annotations. Does not support corrupt PDF documents. Snowbound Software requires that the fonts needed be available on the system. This is a multi-page file format.							
PDF (without plugin)	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-1b – Level B (basic) conformance specification and conforms to PDF v1.7**. JPEG2000 objects within a PDF file require Snowbound Software's optional JPEG2000 license. 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.	59	No	1, 24	.pdf	O	B	Yes

		File	Input Type	Output Bit		Win	Win	Supports Text Read Write
PDF_15	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-1b – Level B (basic) conformance specification and conforms to PDF v1.7**. JPEG2000 objects within a PDF file require Snowbound Software's optional JPEG2000 license. 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. **Note**: Only supported with RasterMaster .NET or RasterMaster DLL. This format is not yet supported in Rastermaster Java.	79	No	1, 24	.pdf	O	B	Yes
PDF_16	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-1b – Level B (basic) conformance specification and conforms to PDF v1.7**. JPEG2000 objects within a PDF file	92	No	1, 24	.pdf	O	B	Yes

		File Type Number	Input Bit	Common Bit	Win Read Ext.	Win Write	Supports Text Search
	require Snowbound Software's optional JPEG2000 license. 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. **Note**: Only supported with RasterMaster .NET or RasterMaster DLL. This format is not yet supported in Rastermaster Java.						
PDF_LZW	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-1b – Level B (basic) conformance specification and conforms to PDF v1.7**. JPEG2000 objects within a PDF file require Snowbound Software's optional JPEG2000 license. 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.	88	No	1, 24	.pdf	O	B	Yes
PHOTOCD	Kodak photo CD format. Supports only 24-bit images. This format contains at least	39	24	No	.pcd			No

		File Type Number	Input Bit	Common Bit	Win Read Ext.	Win Write	Supports Text Search
	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 resid-ual data stored by Huffman encoding.						
PHOTOSHOP	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.	41	1, 4, 8, 24, 32	1, 8, 24, 32	.psd		B	No
PICT	Apple Macintosh bitmap file format . These images may contain vector information such as lines and circles. Only the bitmap portion of data is decompressed. Uses pack bits compression . Supports 1, 2, 3, 4, 8, 16, 24, and 32-bit images.	15	1, 2, 4, 8, 16, 24, 32	1, 4, 8, 24	.pct		B	No
PNG (optional)	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.	43	1, 4, 8, 16, 24, 32	1, 4, 8, 16, 24, 32	.png		B	No
PPT	Microsoft PowerPoint Binary File Format which is the	85	1, 8, 24, 32	No	.ppt	O		Yes

		File Type Number	Input Bit	Common Bit	Win Read Ext.	Win Write	Supports Text Search
	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.						
	The .pptx format is part of a family of open office XML-based formats developed by Microsoft. It is the default documen t 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		
RAST	Sun raster format. Supports 1, 8, 24, and 32-bits. Run length encoded format.	37	1, 8, 1, 8, 24 24	.ras	B	B	No
	The Rich Text Format is a method of encoding formatted text and graphics for easy transfer between applic- ations.				.rft	O	-
SCITEX	The SCITEX format is a proprietary format originated from SCITEX Cor- poration. Gray scale color and CMYK color images. Usually compressed.		60 24, 32	24, 32	.sct	B	B	No
	SEAR CHABL E_ Searchable PDF, also known as "vector PDF" is a regular PDF file that contains searchable text content vs rasterized text.			59 No	No	pdf	O	B
SVG	Scalable Vector Graphics is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation. SVG allows three types of graphic objects: vector graphics, raster graphics, and text.	102	No	24	.svg	-	B	No
	The SCITEX format is a proprietary format	32		1		tga		BB

		File Type Number	Input Bit	Common Bit	Win	Win Read Ext.	Supports Text Write Search
TARGA16	The SCITEX format is a proprietary format originated from SCITEX Corporation.	32	16	24, 32	.tga	B	B	No
TIFF_2D	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. This is a multi-page file format.	17	1	No	.tif	B		No
TIFF_ABIC	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.	46	4, 8	No	.tif	O		No
TIFF_ABIC_BW	TIFF file with Arithmetic Binary encoding. Requires a special ABIC version of our tools. Very popular for check imaging. BW is used f or 1-bit bi-level and TIFF_ABIC is for 4-bit gray scale images. This is a multi-page file format.	47	1	No	.tif	O		No
TIFF_G3_FAX	ANSI baseline Group 3 or Group 4 compression embedded in a TIFF. This is a multi-page file format.	8	1		.tif	B	B	No
TIFF_G4_FAX	ANSI baseline Group 3 or Group 4 compression embedded in a TIFF. This is a multi-page file format.	10	1		.tif	B	B	No
TIFF_G4_FAX_FO	ANSI baseline Group 3	51	1		.tif	B	B	No

		Number				Win	Win	Supports Text Write Search
	or Group 4 compression embedded in a TIFF. This is a multi-page file format.							
TIFF_G4_ FAX_STRIP	ANSI baseline Group 3 or Group 4 compression embedded in a TIFF. This is a multi-page file format.	67	No			B	B	No
TIFF_HUFFMAN	TIFF file compressed using the Huffman compression algorithm. This is a multi-page file format.	7	1			B	B	No
TIFF_JBIG	Standard ANSI baseline JBIG compression embedded in a TIFF. This is a multi-page file format.	66	1			B	B	No
TIFF_JPEG	Standard ansi baseline JBIG compression embedded in a TIFF. This is a multi-page file format.	40	8, 24	8, 24, 32		B	B	No
TIFF_JPEG7	Black and white gray scale format. This is a multi-page file format.	73	1, 8	1, 8		B	B	No
TIFF_LZW	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.	9	1, 4, 8, 24, 32	1, 4, 8, 16, 24, 32		B	B	No
TIFF_PACK	Simple run length encoding algorithm. This is a multi-page file format.	16	1, 4, 8, 16, 24, 32	1, 8		B	B	No
TIFF UNCOMPRESSED	Uncompressed raw binary data . This is a multi-page file format.	0	1, 2, 4, 8, 16, 24, 32	1, 4, 8, 16, 24, 32		B	B	No
UTF-8	UTF-8 is a text format.It is a variable width encoding for the Unicode character set.It may start with the Byte Order Mark of 0xFF FE	38	1	No	.txt	B		No
UTF-16	UTF-16 is a text format.It is a vari-able width encoding for the Unicode character set.It may start with the Byte Order Mark of 0xFE FF.	87	1, 8, 24, 32	No	.txt	B		No
WBMP	Windows file format for wireless devices.	68	1		.wbmp	B	B	No
WINFAX	A simple header with	58	1	No		B		No

		File Type Number	Input Bit	Common Bit	Win Ext.	Win Read	Supports Text Write Search
	CCITT group 3 compression.						
WMF	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.	6	1, 4, 8, 24	1, 4, 8, 16, 24, 32	.wmf	B	B	No
WPG	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.	5	1, 4, 8, 24	1, 4, 8	.wpg	B	B	No
XBM	Xwindows file format which encodes each pixel as an ASCII byte. Only supports 8-bits per pixel.	20	1		.xbm	B	B	No
Xerox_EPS	Encapsulated Postscript for Xerox.	45	1	No	-	B	B	No
XFA	XML Forms Architecture. XFA is an extension to PDF 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. JPEG2000 objects within a PDF file require Snowbound Software's optional JPEG2000 license. Supports some types of Adobe specified PDF annotations. Does not support corrupt PDF documents.| Snowbound Software requires that the fonts needed be available on the system. This is a	59	1, 2, 4, 8, 16, 24, 32	1, 24	.pdf	O		Yes
XLSX	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.	84	1, 8, 24, 32	No	.xls	O	-	Yes
XLSX	The .xlsx format is part of	95	1, 8, 24,	No	.xlsx	O	-	Yes

		File	Input Type	Output Bit		Win	Win	Supports Text Read Write
	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.		32					
XPM	Xwindows bitmap file format stored as ASCII data. Each pixel is stored as an ASCII byte.	35	1, 4, 8		.xpm	B	B	No
XWD	UNIX XWD Raster format. Each pixel is stored as an ASCII byte.	36	1, 4, 8	1, 8, 24, 32	.xwd	B	B	No

Table C.3: File Type Constants listed 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

File Type Number	File Type Name
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
82	HTML
37	RAST
38	ASCII
39	PHOTOCD
40	TIFF_JPEG
41	PHOTOSHOP
42	IMNET
43	PNG
44	GIF_INTERLACED
45	Xerox_EPS
46	TIFF_ABIC
47	TIFF_ABIC_BW
48	DIB
49	MO:DCA_IOCA
51	TIFF_G4_FAX_FO
52	CCITT_G4_FO
53	CCITT_G3_FO
54	FLASHPIX
55	DICOM
56	JEDMICS

File Type Number	File Type Name
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
68	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
85	POWER_POINT
86	DOC
87	RTF
88	PDF_LZW
89	MSG
90	DWG
91	DXF
92	PDF_16
93	DOCX

File Type Number	File Type Name
94	OOXML
95	XLSX
96	ODT
97	ODS
98	ODF
100	PPTX
101	ODP
102	SVG

APPENDIX E: JavaScript API Descriptions

JavaScript API Descriptions

Table 3.8: Supported JavaScript API Descriptions

Name	Returns	Description
`addBookmark(Content, Page)`	Undefined	Adds a bookmark to the page with the content as it's note or tag. This will throw an error if there is already a bookmark on the page.
`addPageToSele`	Undefined	Adds the specified page number to s
`cancelCurrentSearch()`	Undefined	Stops the current search, leaves whatever results have been returned in place. Use `clearSearchResults()` to remove these.
`clearSearchRe`	Undefined	Clears all traces of the current lights,thumbn
`closeTab (tabNumber)`	Integer	Closes tab corresponding to `tabNumber`. Removes the tab from the UI. Switches view to a different tab in its place. Will return an error if this is the last tab.
`collapseAllSticky`	Undefined	Expands or collapses all sticky r
`consolidateAnnotationLayers()`	Undefined	calls the layer when the user clicks on the "C" button in the Layer Manager. When it is called, all the layers, no matter whether they are vis-

Name	Returns	Description
		ible or not, are consolidated into one layer called Master Layer. It is added to the Layer Manager as another layer. The other layers are also still present in the Layer Manager. The Master Layer contains a copy of all the annotations, which is shown on the viewer, i.e. there are double of all annotations. The method returns undefined and does not have any parameters.
copySele	Undefined	Copies the currently selected (in panel) pages to the clipboard (in this context, this is not referring to the system clipboard). Returns true if there were
copySelectionToNewDocument(newId)	Undefined	Copies selection to a new document.
countPagesForDoc	Undefined	Counts the number of pages for the d
cropPageClient(top, left, bottom, right, page)	Undefined	Calls the crop functionality.
cutSelection(delPages,	Undefined	Cuts the currently selected (in pages to the clipboard (in this context, this is not referring to the system clipboard). If delPages is true, the pages are simply deleted and not placed on the clipboard. If showAlert is true, show an alert dialog.Iifpage manip- ulations are disabled (Default = false,) callback is a function that will be called once
cutSelectionToNewDocument(newId)	Undefined	Cuts selection to a new document.
despeckle	Undefined	Despeckles image.
deleteBookmark(page)	Undefined	Deletes the bookmark on the page specified. If no page is specified, it will attempt to remove the current bookmark. If there is no current bookmark, it will throw an error saying that there is no bookmark specified.
editBookmar	Undefined	Opens the edit bookmark dialog for cifi
emailDocument()	Undefined	Brings up the edit bookmark dialog for the spe-

cified page.

Name	Returns	Description
enterPa	Undefined	Puts the viewer back into the default when a guide or select text mode is selected. Mouse Movements will be interpreted
enterGuideMode()	Undefined	Puts the viewer in Guide mode, allowing guides to be moved and locked on the viewer. Mouse movements will be interpreted as guide manipulation actions.
enterSelectTex	Undefined	Puts the viewer in select text mode text by dragging the cursor across any text area of a vector text document. Mouse
exportDocument()	Undefined	Exports the current document with passed in parameters. Mainly uses parameters from vvExportDialog, but values can be passed directly in, bypassing the Dialog Box.
firs	Undefined	Switches to the first page.
fitHeight()	Undefined	Zooms the current page to fit its height to the exact height of the viewing area.
fit	Undefined	Zooms the current page to fit its exact width of the viewing area. It display the
fitWindow()	Undefined	Zooms the current page to fit the entire page in the viewing area.
	Undefined	Rotates the page horizontally along the X axis.
flipY()	Undefined	Rotates the page vertically along the Y axis.
getActiveTab()	Integer	Returns the index of the currently selected tab.
getBrightness()	Integer	Gets the document's brightness to a particular value. Between -125 and 125.
getClientInsta	String	Returns the `clientInstanceId` parameter. The `ClientInstanceId` is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs. This string may be encrypted. Your content handler should implement the encryption and decryption. VirtualViewer HTML5 for .NET does not look at this value at all.
getContrast()	Integer	Gets the document's contrast to a particular value. Between -125 and 125.
getDisplayName()	String	Returns the current display name.
getDocumentId()	String	Returns the `documentId` parameter. The doc -

Name	Returns	Description
		umentId is used to identify the document in the active tab of the VirtualViewer HTML5 for .NET.
getGamma()	Integer	Gets the document's gamma to a particular value. Between -125 and 125.
getPage	Integer	Returns the number of pages in the currently active document. A negative number indicates an error.
getPageNumber()	Integer	Returns the current page number of the page currently being viewed.
getPageProperties	Undefined	Returns an array of objects that entire set of properties for the
getPagePropertyByCaption()	Undefined	Returns the page property for the given caption as configured to be displayed in the Page Properties dialog box.
getPagePropertyByFi	Undefined	Returns the page property for f
getProperty(key, value)	String	Returns the value of an arbitrary document model property to be passed to the content handler.
getZoomPe		Returns the ratio of image's current height on the screen over its original height. Unfor- tunately, this method does not actually return a percentage. To
goToNextPageWithAnn()	Undefined	Goes to the next page that has an annotation.
goToPrevPageWi	Undefined	Goes to the previous page that has
hideImageInfo()	Undefined	Hides the image info dialog box.
init(o	Undefined	Initialize the viewer. Without viewer is initialized without an open document. If the openDoc parameter is true, the viewer attempts to open the current documentId in the viewer. The function "initSpecifiedDocuments" takes a single parameter: "documentIdAndName."

	Returns	Description
		umentId" and "displayName". "documentId" refers to how the document is identified. "displayName" refers to the name that appears over the thumbnail of the document. If no "displayName" is given (meaning it is null), then the document's "documentId" will be used in place of the "displayName."
`initSpecifiedDocuments(documents)`	Undefined	Takes an array of strings and opens all of them as documents each in a tab.
`initV`	Undefined	Initializes the viewer based on the passed in via the URL query. For example: `&doc-`
`invertImage()`	Undefined	Inverts the colors of the current page.
`isDocumentSearc`	Boolean	Returns true if the document is searchable. Returns false if the document is not searchable.
`lastPage()`	Undefined	Switches to the last page.
`nex`	Undefined	Switches to the next page.
`nextSearchResult()`	Undefined	Moves the currently selected search result, switching pages if necessary.
`openInTab(id, newDocument)`	Boolean	Creates a tab for document with id as `id`. Handles the initialization of a new document within a new tab element.
`pageContainsAnnotations()`	Undefined	Called during page selection to return a value of true or false indicating if that page has annotations associated with it.
`pagesWithAnnota`	Undefined	Returns a list of all the pages with annotations.
`pasteSelection(beforePageNum)`	Undefined	Pastes the pages contained on the clipboard into the document.
`previou`	Undefined	Switches to the previous page.
`previousSearchResult()`	Undefined	Moves the currently selected search result, switching pages if necessary.
`printDoc`	Undefined	Prints the current document with parameters. Mainly uses parameters from vvNewPrintDialog, but values can

Name	Returns	Description
printDocument()	Undefined	Will cause the UI to present the server-side printing dialog
printDocument(	Undefined	Will cause the UI to present the client-ing dialog, as was the case in V3.4
redoAnnotation()	Undefined	Redoes the last undo operation.
reloadDocument	Undefined	Reloadsthe document model from discarding any current modifications except for page manipulations. Please note that
removeAllAnnotationIndicators()	Undefined	Removes all the annotation indicators.
removeAl	Undefined	Removes the crop preview on the
removeAnnotationIndicators(pageNumber)	Undefined	Removes an annotation indicator from specified thumbnail.
removeCropOnPag	Undefined	Removes the crop preview on the page.
removeDocumentFromCache()	Undefined	Tells the server to remove a document from the cache.
removePageFromSele	Undefined	Removes the specified pagenumber current
rotateClock()	Undefined	Rotate the current document clockwise 90 degrees.
rotateCo	Undefined	Rotate the current document wise 90
rotateAllPagesBy()	Undefined	Rotates all of the documents pages 0, 90, 180 or 270 (positive or negative) degrees from it's current state.
rotateImageBy(d	Undefined	Rotate the image by 0, 90, 180 or 270 fromit's current unsaved
rotateImageTo(degrees)	Undefined	Rotates the image to 0, 90, 180, or 270 degrees from it's original (saved) rotation. To get to the original position you would call

		`rotateImageTo(0)`
`rotatePageBy(pageNumber,`	Undefined	Rotates the current page 0, 90, 180 or itive or negative) degrees from it's current state. So, you call this twice with 90 degrees as the parameter, the final image will
`rotatePageTo(pageNumber,`	Undefined	Rotates the document 0, 90, 180 or 270

Name	Returns	Description
angle)		degrees absolutely. Thus, if you call this twice with 90 degrees as the parameter, the final image will only be rotated by 90 degrees only. It returns true if the page is rotated successfully. Otherwise, it throws an error.
saveAllDocument	Undefined	Saves all currently opened ing any image manipulations and annotations. When calling saveAllDocuments(), pass the value true as a parameter. This will
saveDocument()	Undefined	Saves the passed in documentId's document including any image manipulations and annotations. If no documentId is passed in, the current document will be saved.
searchDocumentText()	Undefined	Searches for a term between the
sendDocument()	Undefined	Sends the document via the content handler by way of the server. The variable sendDocumentWithAnnotations in config.js determines whether or not annotations are burned in.
setAnnotationsEnabled(enabled)	Boolean	Sets whether or not annotations are enabled. If true, annotations will be enabled. If false, they will be disabled.
setBrightness(value)	Undefined	Sets the document's brightness to a particular value. Between -125 and 125.
setClientInsta	String	Allows the clientInstanceId to be set via JavaScript method. The ClientInstanceId is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs. This string may be encrypted. Your content handler should implement the encryption and decryption. VirtualViewer HTML5 for .NET does not look at this value at all.
setConsolidateLayerNameGenerator()	Undefined	Sets a call-back function to be called when creating a consolidated, master layer. The returned string is used as the filename. It takes in the parameter 'fn'. This is the function to call when exporting.
setContrast	Undefined	Sets the document's contrast to a value. Between -125
setDocumentId(id)	String	Sets the current document id.

Name	Returns	Description
		Note: This method is only supported before installation. Once the viewer is initialized you should use openInTab() to open a new document.
setSendDocumentCompletedHandler (fn)	Undefined	Sets a callback function to be document has
setDocumentIdGenerator (fn)	Undefined	Sets a callback function to be called when creating a new document. Instead of prompting the user for a document id the passed function will be called.
setExportDocument	Undefined	Allows a document name to be the export function is called. Pass a function to this method. That function will be called whenever the user clicks Export. The return value of that function will be sent to the servlet and
setGamma (value)	Undefined	Sets the document's gamma to a particular value. Between -125 and 125.
setImageLoadCompletedHandler	Undefined	Sets a callback function to be called image has finished
setImageLoadRequestedHandler (Function)	Undefined	Sets a callback function to be called when the image has been requested.
setMagnifierPositio	Undefined	Sets the position of the Magnifier.
setPage (page)	Undefined	Switches to the specified page.
setPageManipulationsEnabled	Boolean	Sets whether or not page manipulations are enabled. If true, page
setPageChangeCompletedHandler (function)	Undefined	Sets a callback function to be called when a page has been changed.
setProperty		Sets an arbitrary key/value property pair in the document model to be passed to
setRedactionsEnabled(enabled)	Boolean	Sets whether or not redactions are enabled. If true, redactions will be enabled. If false, they will be disabled.
setRotationCompletedHandler	Undefined	Sets a callback function to be page has

Name	Returns	Description
setSaveDocumentCompletedHandler(fn)	Undefined	Sets a callback function to be called when a document has been saved.
setZoomPercent(percentLevel)	Undefined	Sets the zoom to example, a percentLevel of 75 corresponds
showAboutDialog()	Undefined	Displays the About dialog box.
showAllAnnotationIndicators()	Undefined	Adds indicators to all the pages ann
showAnnotationNaviPanel()	Undefined	Displays the annotation navigation panel when the annotation navigator toggle button is clicked on.
showAnnotationIndicator	Undefined	Add indicator to specific page's functions adds an indicator to the specified
showImageInfo()	Undefined	Displays the image info dialog box.
splitS	Undefined	Splits the VirtualViewer panel and ument in the
showTagAllRedactionsDialog()	Undefined	Redacts all of the current search results, regardless of what page they appear on.
showUploadLocalFileDialog()	Undefined	Shows the Upload Document dialog box.
thumbsWithAnnota	Undefined	Creates a List of thumb boxes of have annotations
toggleCrosshairGuide()	Undefined	Toggle the visibility of the crosshair guide.
toggleH	Undefined	Toggle the visibility ofthe horizontal guide.
toggleImageInfo()	Undefined	Toggles the display of the image info dialog box.
toggleLayerMa	Undefined	Toggles the visibility of the Layer Manager UI.
resetSVGSupport()	Undefined	Toggles SVG support on and off: This will toggle support on and off for the current viewer session as well as implicitly reload the image fromthe server.

Name	Returns	Description
toggleThumbnailPanel	Undefined	Toggles the display of the thumbnail true, show the thumbnail panel. If false, hide the thumbnail panel. If undefined,
toggleVGuide()	Undefined	Toggle the visibility of the vertical guide.
undoAnnot	Undefined	Undoes the last creation of or change
zoomIn()	Undefined	Zooms in to the next level on the current document. The first zoom will fit the document to the window, which may be a large increment. Smaller increments occur after the first zoom. The maxZoomPercent configuration parameter determines how far the page can be zoomed in.
undoSplitS	Undefined	Closes document comparison.
zoomOut()	Undefined	Zooms out to the next level on the current document.
zoomRubbe		Activates the zoom rubber band mode, allowing using the mouse on the currently selected page. When the user clicks, the
zoomToLocation()	Undefined	Zooms to specified percentage level and scrolls the document to a specific location.

Document Methods for Setting, Printing, Exporting, and Saving

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

getDocumentId()

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

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

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

Parameter

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

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

getClientInstan

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

Parameter

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

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

Returns

The `ClientInstanceId` of the current document

setClientInstanceId(id)

This method sets the `ClientInstanceId`. The `ClientInstanceId` is your string to hold whatever information you need. It is often used to hold session or other client specific information that the content handler (Connector) needs. This string may be encrypted. Your content handler should implement

the encryption and decryption. VirtualViewer HTML5 for .NET does not look at this value at all.

Parameter

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

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

Returns

Undefined

setDocumentId(id)

This method sets the current document id.

Note:
This method is only supported before installation. Once the viewer is initialized you should use openInTab() to open a new document.

Parameter

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

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

Returns

Undefined

setSendDocumentCompletedHandler(fn)

This method sets a callback function to be called when a document has been sent.

Parameter

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

Parameter	Type	Description
`fn`	`Function`	The function to call when document has been sent.

Returns

Undefined

setDocumentIdGenerator(fn)

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

Parameter

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

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

Returns

Undefined

setSaveDocumentCompletedHandler(fn)

This method sets a callback function to be called when a document has been saved.

Parameter

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

Parameter	Type	Description
`fn`	`Function`	The function to call when the document has been saved.

Returns

Undefined

setExportDocumentNameGenerator(fn)

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

Parameter

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

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

Returns

Undefined

setDocumentId(id)

This method sets the current document id.

Parameter

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

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

Returns

Undefined

getDisplayNa

This method returns the current display name.

Returns

String

getProperty(key, value)

This method returns the value of an arbitrary document model property to be passed to the content handler.

Returns

String

setProperty(key, value)

This method sets an arbitrary key/value property pair in the document model to be passed to the content handler.

Returns

Void

emailDocument()

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

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

Parameter

The `emailDocument()` method contains the following parameters:

Parameter	Type	Description
`emailFormat`	`String`	exportFormat Either "Original", "PDF" or "TIFF".
`includeTextAnnota`	`Boolean`	Whether or not to include text annotations.
`includeNonTextAnnotations`	`Boolean`	Whether or not to include non-text annotations. *
`includeRedac`	`Boolean`	Whether or not to burn in
`includeRedactionTags`	`Boolean`	IncludeRedactionTags

Parameter	Type	Description
		Whether or not to include redaction tags.
`pageRangeType`	`String`	Either "pages", "complex" or "current".
`pageRangeValue`	`String`	A range of pages numbers to export.
`fromAddress`	`String`	The sender's email address. * Default can be changed in config.js.
`toAddresses`	`String`	The recipient's email address. Default can be changed in config.js.
`ccAddresses`	`String`	Anyone that you would like to CC. Default can be changed in config.js.
`bccAddresses`	`String`	Anyone that you would like to CC. Multiple addresses can be inputed. Seperated by comma. Default can be changed in config.js.
`subject`	`String`	Subject The subject line (title) of the email. Default can be changed in config.js.
`emailBody`	`String`	The message (body) of the email. Default can be changed in config.js.

Returns : undefined

exportDocument()

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

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

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

Parameter

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

Parameter	Type	Description
`exportFormat`	`String`	exportFormat Either "Original", "PDF" or "TIFF".
`fileExte`	`String`	Based on exportFormat. "Original" = state.-getFileExtension(). "PDF" = pdf. "TIFF" = tif..
`includeTextAnnotations`	`Boolean`	Whether or not to include text annotations.
`includeNonTextAnnotations`	`Boolean`	Whether or not to include non-text annotations. *
`includeRedactions`	`Boolean`	Whether or not to burn in redactions.
`includeRedactio`	`Boolean` IncludeRedactionTags	Whether or not to include

Parameter	Type	Description
pageRangeType	String	Either "pages", "complex" or "current".
pageRangeValue	String	A range of pages numbers to export.

Returns

Undefined

GetPageCount()

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

Type

Integer

Returns

The current page count

printDocument()

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

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

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

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

Parameter	Type	Description
printDestination	String	Either "Local", or "Server".
includeTextAnnota	Boolean	Whether or not to include text annotations.
includeNonTextAnnotations	Boolean	Whether or not to include

Parameter	Type	Description
		non-text annotations. *
includeRedactions	Boolean	Whether or not to burn in redactions.
includeRedactionTags	Boolean	IncludeRedactionTags Whether or not to include redaction tags.
pageRangeType	String	Either "pages", "complex" or "current".
pageRangeValue	String	A range of pages numbers to export.
grayScaleChecked	Boolean	Whether or not the image will be in Color or not.
printerName	String	The name of the printer being printed to. Only applies to Network/Server

Returns

Undefined

printDocument()

This method will cause the UI to present the client-side printing dialog, as was the case in V3.4 and earlier.

Returns

Undefined

printDocumentServer()

This method will cause the UI to present the server-side printing dialog.

Returns

Undefined

reloadDocumentModel()

This method reloads the document model from the server discarding any current modifications except for page manipulations. Please note that this does not currently affectannotations.

Returns

Undefined

saveDocument()

This method saves the passed in documentId's document including any image manipulations and annotations. If no documentId is passed in, the current document will be saved. Please see the example below:

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

Parameter

The `sendDocument()` method contains the following parameters:

Parameter	Type	Description
`sync`	`boolean`	Whether to make the request asynchronously or not. Due to legacy browser considerations this should be set to true.
`docId`	`String`	The documentId refering to the document that will be saved. (Optional: If unspecificed the current active document will be used)
`newDocumentId`	`String`	Used to save the current document as a new document in the system. The original document will remain unchanged.

Parameter	Type	Description
		(Optional: Subsequent parameters will be ignored if omitted)
newDispla	S	Used to set the display name of the new document. (Optional:
burnRedactions	Boolean	Used to permanently burn redactions into the new document.
includeRedactionTags	Boolean	IncludeRedactionTags Whether or not to include redaction tags.
burnTextAnnotations	Boolean	Used to permanently burn text annotations into the new document. (Optional: default false)
burnNonTextAnnotations	Boolean	Used to permanently burn non-text annotations into the new document. (Optional: default false)
copyAnnotations	Boolean	Used to copy annotation layers (including redactions) into the new document. (Optional: default false)
saveAsF	S	What the outfile format of the file will be (for now either
pageRangeType	String	Either "pages", "complex" or "current". Only supported when saving to a new document not

Parameter	Type	Description
		supported when saving changes to an existing document
pageRange	S	A range of pages numbers to export. Only supported when saving to a new document not supported

Returns

Undefined

Returns

Undefined

saveAllDocuments()

This method saves all currently opened documents including any image manipulations and annotations. When calling `saveAllDocuments`, pass the value true as a parameter. This will make the call synchronous. This is important when saving multiple documents.

Parameter

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

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

Returns

Undefined

setImageLoadCompletedHandler(Function)

This method sets a callback function to be called when the image has finished loading.

Parameter

The `setImageLoadCompletedHandler(Function)` method contains the following parameter:

Parameter	Type	Description
`Function`	`fn`	The function to call when the image has finished loading.

Returns

Undefined

setImageLoadRequestedHandler(Function)

This method sets a callback function to be called when the image has been requested.

Parameter

The `setImageLoadRequestedHandler(Function)` method contains the following parameter:

Parameter	Type	Description
`Function`	`fn`	The function to call when the image has finished loading.

Returns

Undefined

sendDocument()

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

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

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

Returns

Undefined

removeDocumentFromCache()

This method tells the server to remove a document from the cache with Ehcache 3.1. Ehcache is an open source, standards-based cache that boosts performance, offloads your database, and simplifies scalability.

If using an older version of Ehcache, you may need to make some configuration changes to Ehcache version 3.1. The documentation for Ehcache version 3.1 is available at http://www.ehcache.org/documentation/3.1.

This method has the following dependencies:

The **ehcache.xml** file is included in the **WEB-INF** directory in your build.

The following jar files are included in the **WEB-INF/lib** directory in your build:

lib/ehcache-3.1.1.jar

lib/slf4j-api-1.7.21.jar

lib/slf4j-simple-1.7.21.jar

Use the Servlet action endpoint as shown in the example below:

```
http://-
loc-
alhost:8080/vv/AjaxServlet?action=removeDocumentFromCache&documentId=6
- Pages.tif&clientInstanceId=foo
```

This returns a JSON object with the following members:

status - Either "OK" or "ERROR"

existedInCache - returns true if the document specified was actually cached.

Parameter

The `removeDocumentFromCache(documentId, documentId)` method contains the following parameters:

Parameter	Type	Description
documentId	String	The documentId of the document to be removed.
clientInsta	S	If omitted (i.e. undefined), the current clientInstanceId of VV will be used. If included (i.e. a value or null) will be used in place

Returns

Undefined

Interacting with Document Pages within the Viewer

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

splitScreen()

This method splits the VirtualViewer panel and places a document in the bottom pane.

Parameter

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

Parameter	Type	Description
documentId	String	The id of the document meant for the bottom pane.

Returns

Undefined

undoSplitScreen()

This method closes document comparison.

Returns

Undefined

addPageToSelection()

This method adds the specified page number to the current selection.

Returns

Undefined

closeTab(tabNumber)

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

Parameter

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

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

Returns

Undefined

copyToNewDocument(newDocumentId)

This method copies selection to a new document.

Returns

Undefined

copySelection()

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

Type

boolean

Returns

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

collapseAllStickyNotes()

This method expands or collapses all sticky notes on current page.

Parameters

The `collapseAllStickyNotes()` method contains the following parameters:

Parameter	Type	Description
`collapsed`	`boolean`	If true, all sticky notes will be collapsed.
`repaint`	`boolean`	If true, annotations will be repainted before exit.

Returns

Undefined

countPagesForDocument()

This method counts the number of pages for the specified document.

Returns

Undefined

cutSelection(delPages, showAlert, callback)

This method cuts the currently selected (in thumbnail panel) pages to the clipboard (in this context, this is not referring to the system clipboard). If delPages is true, the pages are simply deleted and not placed on the clipboard. If showAlert is true, show an alert dialog.Iif page manipulations are disabled (Default = false,) callback is a function that will be called once the clipboard object is actually stored by localforage (Optional).

Parameter

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

Parameter	Type	Description
`delPages`	`boolean`	If true, the pages will be deleted and the clipboard will remain unmodified.
`showAlert`	`boolean`	Show an alert dialog if page manipulations are diabled (Default = false.

Parameter	Type	Description
`callback`	`Function`	A function that will be called once the clipboard object is actually stored by localforage (Optional).

Type

boolean

Returns

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

copyToNewDocument(newId)

This method cuts selection to a new document.

Type

String

Returns

Undefined

cropPageClient(top, left, bottom, right, page)

This method calls the crop functionality.

Returns

Undefined

removeAllCrop()

This method removes the crop preview on the page..

Returns

Undefined

removeCropOnPage(page)

This method removes the crop preview on the entire document.

Returns

Undefined

despeckleImage()

This method despeckles the image.

Type

boolean

Returns

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

enterGuideMode()

This method puts the viewer in Guide mode, allowing guides to be moved and locked on the viewer. Mouse movements will be interpreted as guide manipulation actions.

Returns

Undefined

enterPanMode()

This method puts the viewer back into the default pan mode when a guide or select text mode is selected.

Returns

Undefined

enterSelectTextMode()

This puts the viewer in select text mode to select text by dragging the cursor across any text area of a vector text document.

Returns

Undefined

getActiveTab()

This method returns the index of the currently selected tab.

Type

Integer

Returns

The zero-based index of the active tab

getBrightness()

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

Type

Integer

Returns

A value between -125 and 125.

getContrast()

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

Type

Integer

Returns

A value between -125 and 125.

getGamma()

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

Type

Integer

Returns

A value between -125 and 125.

getPageNumber()

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

Type

Integer

Returns

Zero-based page index

getPageProperties()

This method returns a String that represents the entire set of properties for the current page.

Fields

The `getPageProperties()` method contains the following fields:

Fields	Example
`.fieldId`	`documentId`
`.fieldCaption`	`documentId`
`.f`	`ieldValue  MyFile.tif`

Returns

Undefined

getPagePropertyByCaption()

This method returns the page property for the given caption as configured to be displayed in the Page Properties dialog box.

Returns

Undefined

getPagePropertyByFieldId()

This method returns the page property for the given `fieldId`.

Returns

Undefined

hideImageInfo()

This method hides the image info dialog box.

Returns

Undefined

removePageFromSelection()

This method removes the specified page number from the current selection.

Returns

Undefined

showImageInfo()

This method displays the image info dialog box.

Returns

Undefined

toggleImageInfo()

This method toggles the display of the image info dialog box.

Returns

Undefined

toggleCrosshairGuide()

This method toggles the visibility of the crosshair guide.

Returns

Undefined

toggleHGuide()

This method toggles the visibility of the horizontal guide.

Returns

Undefined

toggleVGuide()

This method toggles the visibility of the vertical guide.

Returns

Undefined

resetSVGSupport()

This method toggles support on and off for the current viewer session as well as implictly reload the image from the server.

Returns

Undefined

consolidateAnnotationLayers()

This method calls the layer when the user clicks on the “C” button in the Layer Manager. When it is called, all the layers, no matter whether they are visible or not, are consolidated into one layer called Master Layer. It is added to the Layer Manager as another layer. The other layers are also still present in the Layer Manager. The Master Layer contains a copy of all the annotations, which is shown on the viewer, i.e. there are double of all annotations. The method returns undefined and does not have any parameters.

Returns

Undefined

setConsolidateLayerNameGenerator()

This method sets a call-back function to be called when creating a consolidated, master layer. The returned string is used as the filename. It takes in the parameter ‘fn’. This is the function to call when exporting..

Returns

Undefined

thumbsWithAnnotations()

This method creates a List of thumb boxes of pages that have annotations on them.

Returns

Undefined

showAnnotationIndicator()

This method adds indicator to specific page’s thumbnail. This functions adds an indicator to the specified thumbnail.

Returns

Undefined

removeAnnotationIndicators(pageNumber)

This method removes an annotation indicator from specified thumbnail.

Returns

Undefined

removeAllAnnotationIndicators()

This method removes all the annotation indicators.

Returns

Undefined

showAllAnnotationIndicators()

This method adds indicators to all the pages that have annotations.

Returns

Undefined

showAnnotationNaviPanel()

This method displays the annotation navigation panel when the annotation navigator toggle button is clicked on.

Returns: Undefined

pagesWithAnnotations()

This method returns a list of all the pages with annotations.

Returns

Undefined

goToNextPageWithAnn()

This method goes to the next page that has an annotation.

Returns

Undefined

goToPrevPageWithAnn()

This method goes to the previous page that has an annotation.

Returns

Undefined

redoAnnotation()

This method redoes the last undo operation.

Returns

Undefined

undoAnnotation()

This method undoes the last creation of or change to annotations.

Returns

Undefined

setAnnotationsEnabled()

This method sets whether or not annotations are enabled. If true, annotations will be enabled. If false, they will be disabled.

Returns

Boolean

setRedactionsEnabled()

This method sets whether or not redactions are enabled. If true, redactions will be enabled. If false, they will be disabled.

Returns

Boolean

pageManipulationsEnabled()

This method sets whether or not page manipulations are enabled. If true, page manipulations will be enabled. If false, they will be disabled.

Returns

Boolean

firstPage()

This method switches to the first page.

Returns

Undefined

nextPage()

This method switches to the next page.

Returns

Undefined

pasteSelection(beforePageNum)

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

Parameter

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

Parameter	Type	Description
`beforePageNum`	`Integer`	A zero-based page index specifying where to insert the new pages.

Returns

Undefined

previousPage()

This method switches to the previous page.

Returns

Undefined

lastPage()

This method switches to the last page.

Returns

Undefined

setPageChangeCompletedHandler(function)

This method sets a callback function to be called when a page has been changed.

For example:

```
virtualViewer.setPageChangeCompletedHandler(function(){console.log
("Go to Next Page")})
```

Returns

Undefined

setRotationCompletedHandler(function)

This method sets a callback function to be called when a page has been rotated..

For example:

```
virtualViewer.setRotationCompletedHandler(function(){console.log
("Go to NextPage")})
```

Returns

Undefined

flipX()

This method rotates the page horizontally along the X axis.

Returns

Undefined

flipY()

This method rotates the page vertically along the Y axis.

Returns

Undefined

init(openDoc)

This method initialize the viewer. Without arguments the viewer is initialized without an open document. If the `openDoc` parameter is true, the viewer attempts to open the current `documentId` in the viewer.

Parameter

The `init(openDoc)` method contains the following parameter:

Parameter	Type	Description
`openDoc`	boolean	Whether or not to open a tab after initialization.

Returns

Boolean

initSpecifiedDocuments(documents)

This method takes an array of strings and opens all of them as documents each in a tab.

The method "initSpecifiedDocuments" takes a single parameter: "documentIdAndName"

This parameter is a DocDis Object. DocDis is an Object containing two String variables: "documentId" and "displayNam.e"

"documentId" refers to how the document is identified.

"displayName" refers to the name that appears over the thumbnail of the document.

If no "displayName" is given (meaning it is null), then the document's "documentId" will be used in place of the "displayName."

Parameter

The `initSpecifiedDocuments(documents)` method contains the following parameter:

Parameter	Type	Description
`documents`	String	An array of document ids

Returns

Undefined

initViaURL()

This method initializes the viewer based on the parameters passed in via the URL query. For example:

```
&documentId=foo&clientInstanceId=bar
```

For more information, see setDocumentId and setClientInstanceId.

invertImage()

This method inverts the colors of the current page.

Returns

Undefined

openInTab(id, newDocument)

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

Parameter

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

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

Returns

Undefined

rotateAllPagesBy(pageNumber, angle)

This method rotates all of the documents pages 0, 90, 180 or 270 (positive or negative) degrees from it's current state

Type

Integer

Returns

Undefined

rotatePageBy(pageNumber, angle)

This method rotates the current page 0, 90, 180 or 270 (positive or negative) degrees from it's current state. So, you call this twice with 90 degrees as the parameter, the final image will be rotated by 180. It returns true if the page is rotated successfully. Otherwise, it throws an error.

Type

Integer

Returns

Undefined

rotatePageTo(pageNumber, angle)

This method rotates the document 0, 90, 180 or 270 degrees absolutely. Thus, if you call this twice with 90 degrees as the parameter, the final image will only be rotated by 90 degrees only. It returns true if the page is rotated successfully. Otherwise, it throws an error.

Type

Integer

Returns

Undefined

rotateClock()

This method rotates the current document clockwise 90 degrees.

Returns

Undefined

rotateCounter()

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

Returns

Undefined

rotateImageBy(degrees)

This method rotate the image by 0, 90, 180 or 270 degrees from it’s current unsaved rotation.

Type

Integer

Returns

Undefined

rotateImageTo(degrees)

This method rotates the image to 0, 90, 180, or 270 degrees from it's original (saved) rotation. To get to the original position you would call `rotateImageTo(0)`.

Type

Integer

setBrightness(value)

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

Returns

Undefined

setContrast(value)

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

Returns

Undefined

setGamma(value)

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

Returns

Undefined

setPage(page)

This method switches to the specified page.

Parameter

The `setPage(page)` method contains the following parameter:

Parameter	Type	Description
page	Integer	Zero-based page index.

Returns

Undefined

showAboutDialog()

This method displays the About dialog box.

Returns

Undefined

showUploadLocalFileDialog()

This method shows the Upload Document dialog.

Returns

Undefined

toggleLayerManager()

This method toggles the display of the layer manager.

Returns

Undefined

toggleThumbnailPanel(show)

This method toggles the display of the thumbnail panel.

Parameter

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

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

Returns

Undefined

addBookmark(content, page)

This method adds a bookmark to the page with the content as it's note or tag. This will throw an error if there is already a bookmark on the page.

Parameter

The `addBookmark(content, page)` method contains the following parameters:

Parameter	Type	Description
`content`	String	Sets the bookmark content.
`page`	Integer	Zero-based page index.

Returns

Undefined

deleteBookmark(page)

This method deletes the bookmark on the page specified. If no page is specified, it will attempt to remove the current bookmark. If there is no current bookmark, it will throw an error saying that there is no bookmark specified.

Parameter

The `deleteBookmark(page)` method contains the following parameter:

Parameter	Type	Description
`page`	Integer	Zero-based page index.

Returns

Undefined

editBookmark(

This method opens the edit bookmark dialog for the specified page.

Parameter

The `editBookmark(page)` method contains the following parameter:

Parameter	Type	Description
`page`	Integer	Zero-based page index.

Returns

Undefined

Adjusting the Size of the Window

This section describes the methods for adjusting the size of the window.

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

fitWidth()

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

Returns

Undefined

fitHeight()

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

Returns

Undefined

fitWindow()

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

Returns

Undefined

setMagnifierPosition()

This method Sets the position of the Magnifier.

Parameter

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

Parameter	Type	Description
x	Integer	The X position of the Magnifier.
y	Integer	The Y position of the Magnifier.

Returns

Undefined

getZoomPerc

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

Type

Float

Returns

The zoom ratio

setZoomPercent(percentLevel)

This method sets the zoom to percentLevel directly. For example, a percentLevel of 75 corresponds to 75%.

Returns

Undefined

zoomIn()

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

Returns

Undefined

zoomOut()

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

Returns

Undefined

zoomRubberband()

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

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

Returns

Undefined

zoomToLocation()

This method zooms to specified percentage level and scrolls the document to a specific location.

Parameter

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

Parameter	Type	Description
zoom	Integer	The new zoom percentage, in absolute terms.
x	Integer	The new x position, in the zoomed coordinate space.
y	Integer	The new y position, in the zoomed coordinate space.

Returns

Undefined

Searching

This section describes the method for searching.

cancelCurrentSearch()

This method stops the current search, leaves whatever results have been returned in place. Use clearSearchResults() to remove these.

Returns

Undefined.

clearSearchResults()

This method clears all traces of the current search (highlights,thumbnails, etc).

Fields

The `clearSearchResults()` method contains the following fields:

Parameter	Type	Description
`state`	String	The state of the document.

Type

Undefined

Returns

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

isDocumentSearchable()

This method determines if the current document is searchable.

Type

boolean

Returns

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

nextSearchResult()

This method moves the currently selected search result, switching pages if necessary.

Returns

Undefined.

previousSearchResult()

This method moves the currently selected search result, switching pages if necessary.

Returns

Undefined.

showTagAllRedactionsDialog()

This method adds redaction tags to the search and redact results

Returns

Undefined.

searchText()

This method searches for a term between the pages specified.

Fields

The `searchText()` method contains the following fields:

Parameter	Type	Description
term	String	The string to search for in the document.
firstPage	Integer	A zero-based page index bracketing the start of the range of pages to search.
lastPage	Integer	A zero-based page index bracketing the end of the range of pages to search.

Returns

Undefined

APPENDIX F: Working with the ContentHandler

Connecting to Your Document Store

VirtualViewer HTML5 for .NET comes with a sample file content handler that helps you connect VirtualViewer HTML5 for .NET to your file system. Snowbound Software has content handlers available to connect to Alfresco, FileNet P8, Documentum Webtop, Pega Systems You can also create your own custom connector or use Snowbound Professional Services to create a custom content handler for you.

Note:
VirtualViewer HTML5 for .NET includes a sample content handler that connects to the server's File System. This code sample is provided as a starting point to integrate VirtualViewer HTML5 for .NET with your document storage. The content handler sample is not production quality. You should add error checking and permissions checking at the very least before releasing this into production.

What is the Content Handler?

The VirtualViewer HTML5 for .NET content handler is a DLL that the server will call on to perform various actions concerning the retrieval and storage of content. By default, the VirtualViewer server uses the sample content handler that Snowbound Software provides, **FileContentHandler.dll**, as its content handler, which reads and writes to a file system location. You can find this sample content handler at `VirtualViewerNetHTML5\Sample-Code\.Net Content Handler`. It displays files from the `C:/imgs` directory. You are encouraged to use this as a starting point for writing your own custom DLL 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.

VirtualViewer HTML5 for .NET Content Handlers

VirtualViewer HTML5 for .NET has three available sample content handlers for general use. Both the VirtualViewer and the URL content handlers are shipped with the standard VirtualViewer HTML5 for .NET package.

FileContentHandler.dll

The VirtualViewer HTML5 for .NET content handler is the sample content handler shipped with VirtualViewer HTML5 for .NET. This content handler supports annotations and will load images from a designated directory on your server. All configurations for this content handler are done before shipping.

Note:
The filecontentHandler will handle byte arrays, but it does not support input streams.

DEPRECATED - URLContentHandler.dll

DEPRECATED -The URL content handler allows the viewing of web images by simply input- ting the URL of that image into the `documentID`. To enable the URL con- tent handler, simply update your `contentHandler` parameter in your content server's `web.config` file to match the following:

```
<add key="contentHandler" value-
="C:/Ine-
etpub/VirtualViewer/VirtualViewerNetHTML5/bin/URLContentHandler.dll"/>
```

Please note the following:

- The URL content handler does not support annotation saving.
- Documents in your images directory can still be viewed without annotations with this content handler.

It displays files from the `./Sample-Documents` directory that is installed with VirtualViewer HTML5 for .NET

Defining a Custom Content Handler

To tell the server to use a custom content handler in the **web.config** file, the `useDllContentHandler` parameter must be present and set to true. Snowbound provides the source code for the file content handler in the installation. Look for**filecontenthandler.cs**. Use this code as a starting point when creating a custom solution that fits into your environment using your security, repository, and error messages.

The VirtualViewer HTML5 for .NET server will then use the content handler DLL that is specified by the `contentHandler` parameter. For example:

```
<add key="useDllContentHandler" value="true" />
<add key="contentHandler" value="filePathToDLL" />
```

Please open up the filecontenthandler.cs when reading the following sections on the interface so you can follow along in detail.

VirtualViewerNetContentHandlerInterface

This interface defines methods for retrieving content for VirtualViewer HTML5 for .NET. Most of the methods take in a single input parameter, which is an instance of the class `ContentHandlerInput`, a hash table 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 a hash table which contains the data required to complete the method.

Authentication

The authentication is added in the content handler in the get and save method implementations, including getDocumentContent, validateCache, and saveDocumentContent.

VirtualViewer passes along any cookies that are associated with the html page. This mechanism is how we support single sign on (SSO). Additionally, if the HTTP session ID is not part of the cookie string, then VirtualViewer HTML5 for .NET will automatically add the HTTP session ID to the cookie string. This way, the content handler has the information it needs to verify the current user is authorized to view or save the current document.

If you need to pass more authentication information to your custom content handler, you can use the ClientInstanceId to pass it encrypted in whatever way you like. You have to decrypt it before using it.

When customizing the VirtualViewer content handler to connect to your document storage, you may need to request or store authentication tokens as part of the process.

You can store the tokens in the session object within the content handler. Use the `HttpServletRequest` session object in the content handler to achieve this.

The user can get a handle to `HttpServletRequest` session object in the content handler by using this line of code:

Example 1.9: Getting a Handle to `HttpServletRequest` Session Object in the Content Handler

```
HttpServletRequest request = (HttpServletRequest) input.get(Con-
tentHandlerInput.KEY_HTTP_SERVLET_REQUEST);
```

The user can then get or set session attributes:

Example 1.10: Getting or Setting Session Attributes

```
request.getSession().getAttribute(arg0);
request.getSession().setAttribute(arg0, arg1)
```

Single Sign On (SSO)

Single sign on (SSO) related information can be stored in the `ClientInstanceId` parameter or more often in the HTTP session ID. VirtualViewer should pass along any cookies that are associated with the HTML page that contains VirtualViewer. This mechanism is how we support single sign on (SSO). Additionally, if the HTTP session ID is not part of the cookie string, then Virtual Viewer will automatically add the HTTP session ID to the cookie string.

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 `ICacheValidator` in addition to `IVirtualViewerNetContentHandlerInterface` .

CacheValidator Method Detail

`validateCache`

```
public ContentHandlerResult validateCache (Con-
tentHandlerInput input)
throws com.s-
nowbound.snapserv.servlet.FlexSnapSIAPIException
```

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

Parameters

A `ContentHandlerInput` object containing the following data:

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

Please follow the filecontenthandler.cs sample code.

1. Enable the following line

   ```
   public class DllContentHandler :
   IVirtualViewerNetContentHandlerInterface, IVir-
   tualViewerNetSaverInterface, ICacheValidator
   ```

2. Find

   ```
   public ContentHandlerResult validateCache(Con-
   tentHandlerInput input)
   ```

3. Update `validateCache`, put the custom content handler in the method.

Event Notification and Handling

`eventNotification`

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

```
public ContentHandlerResult eventNotification (Con-
tentHandlerInput input) throws FlexSnapSIAPIException
```

Implement this content handler method to receive event notifications.

The `eventNotification` method is called whenever a document is successfully retrieved from the content server's internal cache.

The following auditing events trigger event notification:

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

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

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

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

The following are the details for each event:

Parameters

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

Key	Type	Description
`"KEY_CLIENT_INSTANCE_ID"`	`String`	One of the `VALUE_EVENT_*` values
`"VALUE_EVENT_PAGE_"`	`String`	The event being logged is a
`"KEY_EVENT_PAGE_REQUESTED_NUMBER"`	`String`	The page number requested (zero-based).
`"VALUE_EVENT_SAVE_"`	`String`	The event being logged is a
`"KEY_EVENT_SAVE_ANNOTATION_LAYER_NAME_BASE*"`	`String`	The base name of the keys containing the layer names. There will be one of these for each layer. For example:

Key	Type	Description
		KEY_EVENT_SAVE_ANNOTATION_LAYER_NAME_BASE0
"VALUE_EVENT_P	String	The event being logged is a
"KEY_EVENT_PRINT_PAGE_NUMBERS"	String	The page range being printed, in the format '0-4'
"VALUE_EVENT_EX	String	The event being logged is a
"KEY_ANNOTATION_ID"	String	The name of the annotation layer.

Returns

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

Extracting Parameters from ContentHandlerInput

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

The first method is set 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();
```

There are two ways to pass the `clientInstanceId` from the client to the server and ultimately to the content handler:

1. The client will try to read a HTTP cookie named `clientInstanceId`. If you set a cookie named `clientInstanceId` to the value you wish to pass, it will be passed to the content handler by way of the server.
2. If the cookie cannot be read, the client will fall back to reading the `cli-`

entInstanceId from the URL. It can be passed in the URL string as a standard parameter keypair. For instance:

```
http://servername/VirtualViewerNetHTML5/
index.html?documentId=filename.jpg&clientInstanceId=
"username=user:password=123"
```

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

Table 4.9: Method Summary

Method	Description
getAnnotationContent()	Returns the annotation data for a given layer.
getAnnotatio	Returns the name of the annotation
getAnnotationLayers()	Returns the names of all annotation layers.
getAnnotationProperties()	Returns the properties for a given annotation layer.
getBookmarkContent()	Returns the XML data for bookmarks.
getClientInstanc	Returns the value of the cli-
getDocumentContent()	Returns the data of the document.
getDocumentFile()	For future use.
getDocumentId()	Returns the name or ID of the document.
getHttpCont	Returns the standard Httpcontext

The second method is by explicitly naming the key in the input hash table. For example, to retrieve the same values as the previous example, you would do the following:

```
String clientId = input.KEY_CLIENT_INSTANCE_ID;
String documentId = input.KEY_DOCUMENT_ID;
```

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

Table 4.10: 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
"KEY_ANNOTATION_LAYERS"	AnnotationLayer[]	The information for all annotation layers.
"KEY_ANNOTATION_	Hash	The properties for an annotation
"KEY_BOOKMARK_CONTENT"	byte[]	The XML data for bookmarks.
"KEY_CLIENT_	S	Value of the clientInstanceId
"KEY_DOCUMENT_CONTENT"	byte[]	The data of the document.
"KEY_DOCUMENT_		For
"KEY_DOCUMENT_ID"	String	The name or ID of the document.
"KEY_HTTP_SER		The HttpServletReque
"KEY_MERGE_ANNOTATIONS"	boolean	Indicates if annotations were burned in or not.

Populating Parameters for ContentHandlerInput

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

```
input.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`.

The required data for each method should be put into the `ContentHandlerResult` return object with the `result.key = value` with the desired existing key.

```
result.KEY_DOCUMENT_ID_TO_RELOAD = "test_b.pdf";
```

Table 4.11: Property Descriptions

Property	Description
`"DOCUMENT_ID_TO_RELOAD"`	The `documentId` to load after a save is made.
`"ERROR_MESSAGE"`	The error message if there is an error.
`"KEY_ANNOTATION_CONTENT"`	The annotation data for a given layer.
`"KEY_ANNOTATION_NAMES"`	The names of all annotation layers.
`"KEY_ANNOTATION_PROPERTIES"`	The properties for a given annotation layer.
`"KEY_AVAILABLE_`	The array of `documentId`'s for `avail-`
`"KEY_BOOKMARK_CONTENT"`	The XML data for bookmarks.
`"KEY_DOCUMENT_`	The data ofthe
`"KEY_EXTERNAL_`	Returns a list of `ExternalReference` objects.

`REFERENCE_CONTENT_ELEMENTS"`	To implement external references in your content handler, include references to the ExternalReference class in your code.

"VOID"	Used for null or void returns.

Document Notes Methods

The `getNotesPermissions()` and `getNotesTemplates` methods in FileContentHandler.java are used for Document Notes.

The `getNotesPermission` method must return a ContentHandlerResult containing a permission level. The default is PERM_DELETE. This can be kept as .

The `getNotesTemplates` method must return a ContentHandlerResult. This is never read. It can be left as returning a new (empty) ContentHandlerResult().

To add Note Templates to this method, add the following code:

Example 1.11: Adding Note Templates

```
Vector<NotesTemplate> vTemplates = new Vector<NotesTemplate>();
NotesTemplate template1 = new NotesTemplate("Sample","This is a
sample");
vTemplates.add(template1);
result.put(ContentHandlerResult.KEY_NOTES_TEMPLATES, vTemplates);
```

Sparse Document Support

The Sparse Document feature allows the content handler to return only a partial document. The content handler returns the specific pages that are needed instead of the entire document.

The Sparse Document feature is only intended for multi-page documents that are stored as single pages in the document repository. It allows for the retrieval of arrays of pages instead of the entire document. This improves download times as we no longer have to retrieve the entire document.

The Sparse Document feature does not support use cases for single page documents. It supports only multi-page documents stored as individual single pages.

Use the following `ContentHandlerInput` properties as inputs to `getDocumentContent`:

Property	Description
	"KEY_SPARSE_PAGE_ INDEX"

Sets the page that has to be returned and reques- ted.

"KEY_SPARSE_PAGE_COUNT"	Sets the suggested total number of pages to return starting at that index. For example, VirtualViewer can request 10 pages starting at the second page.

Set your content handler on your server to return a specific number of pages to send to the client on initial load and subsequent page requests that are different from VirtualViewer's requested page count.

The following is an example of the process for the Sparse Document feature:

1. Load a 500 page document for the first time.
2. VirtualViewer requests the first page from the content handler (KEY_SPARSE_PAGE_INDEX) as well as a hint as to how many subsequent pages VirtualViewer thinks it may need shortly(KEY_SPARSE_PAGE_COUNT). For this example, VirtualViewer asks for 10 pages.
3. The first 10 pages of the document are sent to the VirtualViewer client and cached if caching is enabled. Those specific pages are not requested again. The rest of the pages in the document are not requested until they are needed.
4. Select the thumbnail or jump-to-page for page 50.
5. VirtualViewer displays pages 50-60 which is 10 pages from 50, the requested page.

The following is a code example showing how to use the Sparse Document feature. Please note that this is intended to be an example to use for your own purposes and is not intended for production use.

Example 1.12: Sparse Document

```
// This is an example of how to use VV's "Sparse Document" mech-
anism. It is not intended for production use.

public ContentHandlerResult getDocumentContent(ContentHandlerInput
input)
{
	String key = input.getDocumentId();
	int pageNumber = input.getSparseRequestedPageNumber();
	int pageCount = input.getSparseRequestedPageCount();
	Vector documentPages = new Vector();
	for(int i = 0; i < pageCount; i++)
	{

		//getPageData is a standin for some operation that is
		fast for a few pages
		//but slow when called for hundreds of pages
		byte[] pageData = getPageData(key, pageNumber + i);
		documentPages.addElement(pageData);
	}
	//getTotalPagesInDocument is another example just for this
	snippet
	int totalDocumentPageCount = getTotalPagesInDocument(key);
	ContentHandlerResult result = new ContentHandlerResult();
```

```
result.put(result.KEY_DOCUMENT_SPARSE_ELEMENTS, doc-
umentPages);
result.put(result.KEY_DOCUMENT_SPARSE_PAGE_INDEX,
pageNum- ber);
result.put(result.KEY_DOCUMENT_SPARSE_RETURN_PAGE_COUN
T, doc- umentPages.size());
result.put(result.KEY_DOCUMENT_SPARSE_TOTAL_PAGE_COUNT
, totalDocumentPageCount);
return result;
```

Content Handler Methods

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

Table 4.12: VirtualViewerNetContentHandlerInterface

Return Value	Method
ContentHandlerResult	`deleteAnnotation(ContentHandlerInput input)` Returns the content for the specified annotation key in the form of a byte array.
ContentHandlerResult	`eventNotification(ContentHandlerInput input)` Implement this content handler method to receive event notifications.
ContentHandlerResult	`getAnnotationContent(ContentHandlerInput input)` Called when the client has requested to delete the specified annotation layer.
ContentHandlerResult	`getAnnotationNames(ContentHandlerInput input)` Returns an array of annotation object names for the specified `clientInstance` and `doc-`

Retur	Method
	umentKey
ContentHandlerResult	getDocumentContent(ContentHandlerInput input) Returns the content for the specified content key in the form of a byte array.
	init(HttpContext context, NameValueCollection initParams)

VirtualViewerNetSaverInterface

VirtualViewerNetContentHandlerInterface extends VirtualViewerNetSaverInterface

ContentHandle	saveAnnotationContent(Con-
ContentHandlerResult	saveDocumentComponents(ContentHandlerInput input)
ContentHandle	saveDocumentContent(Con-

This section describes the VirtualViewerNetContentHandlerInterface methods.

deleteAnnotation

```
public ContentHandlerResult deleteAnnotation (ContentHandlerInput input)
```

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

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Value	Description
"KEY_CLIENT_INSTANCE_ID"	String	Value of the `clientInstanceId` client parameter.
"KEY_DOCUMENT_	S	The name or ID of the
"KEY_	String	The name of the annotation layer.

Key	Value	Description
ANNOTATION_ID"		

Returns

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

getAnnotationContent

`public ContentHandlerResult getAnnotationContent (ContentHandlerInput input)`

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

Note:
Currently, caching is done for the document and not for the annotation. The client will request an annotation even if the document is cached.

Example 1.13: getAnnotationContent Method

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

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
"KEY_CLIENT_ INSTANCE_ID"	String	Value of the `clientInstanceId` client parameter.

		Description
"KEY_DOCUMENT_ID"		The name or ID ofthe
"KEY_ANNOTATION_ID"	String	The name of the annotation layer.

Returns

A `ContentHandlerResult` object containing the following data:

Key	Type	Description
"KEY_ANNOTATION_CONTENT"	byte[]	The annotation data for a given layer.

---

getAnnotationNames

`public ContentHandlerResult getAnnotationNames (ContentHandlerInput input)`

Called to request an array of annotation object names for the specified `clientInstance and documentId array`.

Example 1.14: getAnnotationNames

```
      ContentHandlerResult


      clientInstanceId =
      input.getClientInstanceId();
String[] arrayNames = new String[2];
arrayNames[0] =
"layerOne";
arrayNames[1] =
"layerTwo";
ContentHandlerResult result = new ContentHandlerResult();
result.Add(ContentHandlerResult.KEY_ANNOTATION_NAMES,
```

Parameters

A `ContentHandlerInput` object containing the following data:

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

Key	Type	Description
"KEY_DOCUMENT_ID"	Typ	The name or ID ofthe

Returns

A `ContentHandlerResult` object containing the following data:

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

---

getAnnotationProperties

getAnnotationProperties

```
public ContentHandlerResult getAnnotationProperties
(ContentHandlerInput input)
```

Called to request the properties for a specified annotation layer in the form of a hash table.

Parameters

A `ContentHandlerInput` object containing the following data:

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

Returns

A `ContentHandlerResult` object containing the following data:

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

---

getDocumentContent

`public ContentHandlerResult getDocumentContent (ContentHandlerInput input)`

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.

Note:
The filecontentHandler will handle byte arrays, but it does not support input streams.

Example 1.15: getDocumentContent Method

```
public ContentHandlerResult getDocumentContent
(ContentHandlerInput input)
{
String clientInstanceId = input.getClientInstanceId();
String key = input.getDocumentId();
// Code to retrieve document goes here
ContentHandlerResult result = new ContentHandlerResult();
result.Add(ContentHandlerResult.KEY_DOCUMENT_CONTENT, documentData);
return result;
}
```

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
`"KEY_HTTP_CONTEXT"`		The standard `HttpContext` data.
`"KEY_CLIENT_`	S	Value of the `clientInstanceId` cli-
`"KEY_DOCUMENT_ID"`	`byte[]`	The contents of the document.

Returns

A `ContentHandlerResult` object containing the following data:

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

Key	Type	Description
CONTENT"		

init

```
public void init(HttpContext context, NameValueCol-
lection initParams)
```

Performs any necessary configuration tasks.

Parameters

`initParams` - The Server config object for the VirtualViewer HTML5 for .OET content server.

Returns

void

VirtualViewerNetSaverInterface Method Detail

This section describes the VirtualViewerNetContentHandlerInterface methods.

saveAnnotationContent

```
public ContentHandlerResult saveAnnotationContent (Con-
tentHandlerInput input)
```

Called to save an annotation layer.

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
"KEY_HTTP_ CONTEXT"		The standard `HttpContext` data.
"KEY_CL IENT_	S	Value of the `clientInstanceId`
"KEY_DOCUMENT_ID"	String	The name or ID of the document.
"KEY_ANNOTATION_	String	The name or ID ofthe annotation
"KEY_ANNOTATION_	byte[]	The annotation data for a given

Key	Type	Description
CONTENT"		layer.

Returns

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

saveDocumentComponents

```
public ContentHandlerResult saveDocumentComponents
(ContentHandlerInput input)
```

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

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

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

Parameters

A `ContentHandlerInput` object containing the following data:

Key	Type	Description
"KEY_HTTP_ CONTEXT"		The standard `HttpContext` data.
"KEY_CL IENT_	S	Value of the `cli- entInstanceId` client para- meter.
"KEY_DOCUMENT_ ID"	String	The name or ID of the document.
"KEY_DOCUMENT_	byte[]	The data ofthe
"KEY_ ANNOTATION_ LAYERS"	AnnotationLayer []	The information for all annotation layers.
"KEY_BOOKMARK_	byte[]	The XML datafor

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 1.16: Key_Annotation_Layers

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

Returns

A ContentHandlerResult object containing the following data:

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