

VirtualViewer® 4.13 (and older)

New Features and Corrected Issues

October 2, 2018

Contents

Important Phone Numbers and Links	4
Filing a Support Ticket.....	4
Release Notes and Product Manuals	4
Important Information.....	4
Important Notices.....	5
VirtualViewer 4.13 Release Notes	5
New Features	5
Save Default Choices for Document Dialogs.....	5
Updated Search	6
Configurable Highlight Colors for Search.....	8
New Callbacks	8
Configuration to Disable User Preferences.....	9
Dynamic Debug Logging	9
Simple Logging Facade (Java)	10
Common Logging Facade (NET)	10
Fixes and Changes	10
Sticky note updates.....	10
enableOcr configuration fixed	11
PDF signature printing issue.....	11

Misc. Fixes/Changes:	11
New Features 4.12	12
Video.....	12
Supported formats.....	12
User preferences.....	12
Supported features with video	13
Set document display name API.....	13
Load document model asynchronously	13
Highlight annotation button currently in use.....	13
Adjust toolbars intelligently based on how many buttons are visible.....	13
Redaction Navigation	13
Page Manipulation with Bookmarks	14
Cache-seeding Support (Java)	14
Fixes and changes	15
VirtualViewer initialization API are easier to work with.....	15
Preserve document scroll when zooming	15
Preserve document scroll and zoom when switching between tabs	15
disableUploadDocument parameter moved to server.....	15
Removed sample content handlers from VirtualViewer distributable	15
InputStreamContentHandler	16
FileAndURLRetriever	16
MergedImageContentHandler	16
MultiContentElementContentHandler.....	16
Public print API.....	16
New Features 4.11	18
Document Compare	18
Callback Event Manager	20
Sticky Note Background Color Support.....	21
User Preference Option for Default Thumbnail Tab	21
Copy Annotations Across Documents.....	21

VirtualViewer 4.13 (and older) New Features and Corrected Issues

Get Page Dimensions.....	22
Require.js	22
Thumbnail name customization.....	23
Changing the Document display name	23
VirtualViewer 4.11 Bug Fixes	25
New Features of VirtualViewer 4.10	26

Important Phone Numbers and Links

For the most current information, please contact Snowbound Sales at

Telephone: 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)

Filing a Support Ticket

<https://snowboundsupport.force.com/SupportPortal/CommunityLogin?>

Release Notes and Product Manuals

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

Important Information

- Redactions may shift for PDFs where a document is redacted, rotated, then saved and then exported to PDF. We are handling this as a highly critical issue and a point release (VV 4.11.x) will shortly be issued.
- Redaction Tags are not burned into exported PDF documents. Also to be fixed in a upcoming point release
- VirtualViewer for Java now supports only JRE 1.7+. 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 because they are unnecessary:
 - defaultByteSize
 - tiffByteSize
 - jpegByteSize

VirtualViewer 4.13 (and older) New Features and Corrected Issues

Important Notices

We have added support for JRE 1.7 in this version of VirtualViewer through the end of 2019. JRE 1.6 is only supported under special arrangements. Given the accelerated update release schedule by Oracle, we strongly recommend that customers upgrade to the latest JRE (currently Java 8) as quickly as possible. Newer JRE's generally support older Java applications without any modification. However features introduced in newer JRE's will not work for older JRE's.

VirtualViewer 4.13 Release Notes

New Features

Save Default Choices for Document Dialogs

Users may now save custom default choices for the Save As, Export, Copy to New, Cut to New, Print, and Email dialogs.

For instance, a user's workflow may demand that all documents be exported as TIFFs. Previously, the user would have to find the Format section in the Export dialog and click the TIFF radio button for every export.

Now, the user can fill out the dialog with their preferred default choices and then click the button labeled Save Preferences in the bottom left of the dialog. When the user opens the Export dialog again, the form will be filled out with their saved defaults.

How to Use

To use this new feature, a user modifies the form choices in a dialog and saves those choices as the new default. For instance, they may choose to set defaults in the print dialog. The user opens the print dialog, and chooses the options to use going forward.

Clicking the Save Preferences button will save the user's choices. The dialog will still open normally, and the user may still change options normally. The options that are selected immediately on opening the dialog will now be the user's custom defaults.

VirtualViewer 4.13 (and older) New Features and Corrected Issues

Technical Details

Data will be stored in the browser's local storage, through the localforage library, so the preferences will persist across sessions of VirtualViewer on the same browser. Radio buttons and checkboxes will be stored; free text fields and page range fields will not have any defaults stored.

New Configuration Options

No configuration is necessary to enable this feature, but there are new configuration options to pre-set certain dialog defaults.

- `vvConfig.includeRedactions` The "Burn Redactions (Permanent)" checkbox will burn redactions into an image. If this configuration item is true, "Burn Redactions (Permanent)" will be checked by default.
- `vvConfig.includeRedactionTags` The "Include Redaction Tags" checkbox will write redaction tags onto the redactions on an image. If this configuration item is true, "Include Redaction Tags" will be checked by default.
- `vvConfig.includeDocumentNotes` The "Include Document Notes" checkbox will include the document notes in the exported, saved, printed, or copied document. If this configuration item is true, "Include Document Notes" will be checked by default.
- `vvConfig.includeWatermarks` The "Include Watermarks" checkbox will include added watermarks in the exported, saved, printed, or copied document. If this configuration item is true, "Include Watermarks" will be checked by default.

Updated Search

VirtualViewer now supports document searches and OCR on Virtual Documents, Sparse Documents, and compound documents. Search will also return correct results on documents whose pages have been manipulated and that have not yet been saved; previously, it would use the server version of a document, so could return results for a deleted page.

Pattern search is now supported on annotations. VirtualViewer may search annotation text, tags, and notes for social security numbers, telephone numbers, credit card numbers, and email addresses.

The user interface of the search tab has been updated with new button images, styles, and an adjusted layout.

VirtualViewer 4.13 (and older) New Features and Corrected Issues

The search API remains largely the same, with a new addition:

Unchanged API:

- `virtualViewer.cancelCurrentSearch()` stops the current search, and displays any already-returned results.
- `virtualViewer.clearSearchResults()` clears the current search, removing highlights from the document and thumbnails from the search panel.
- `virtualViewer.nextSearchResult()` advances the currently selected search result, switching pages if necessary.
- `virtualViewer.previousSearchResult()` moves the currently selected search result to the previous match, switching pages if necessary.
- `virtualViewer.isDocumentSearchable()` returns true if the document is searchable. It returns false if the document is not searchable
- `virtualViewer.searchText(searchTerm, firstPage, lastPage, skipOcrPrompt)` launches a search through the current document's text for the given search term. This search is performed on the server, and may perform OCR if the document has no text, OCR is enabled, and the user consents. A progress bar will appear when search is launched, as document search is performed asynchronously and in batches: a small batch of pages will be searched and a new batch sent to the server when the previous batch is returned.
 - `searchTerm` {String} The word or words to search for. Set case sensitivity in your configuration file.
 - `firstPage` {Number} Optionally define the start of a region of the document to search. This is 0-indexed, and the default is 0.
 - `lastPage` {Number} Optionally define the end of a region of the document to search. This is a 0-indexed, non-inclusive value. The default is the length of the document.
 - `skipOcrPrompt` {Boolean} If this parameter is set, document search will not prompt the user before using OCR, but will go ahead and use it if necessary and if OCR is enabled.
 - returns undefined

New API:

- `virtualViewer.searchAnnotationText(searchTerm)` launches a search through every annotation on the current document for the given search term. Searched annotation text includes annotation notes, text content, and tags. If no search term is provided and there is a search pattern currently selected in the search tab, a pattern search through the annotations will be launched.

VirtualViewer 4.13 (and older) New Features and Corrected Issues

- `searchTerm {String}` The word or words to search for. Annotation search is case-insensitive.
- Returns undefined

Configurable Highlight Colors for Search

Two new configuration parameters allow color customization for search. When a search is completed, all search results are highlighted in an orange color on the document; the current search result in focus is highlighted a light yellow.

The first option, `vvConfig.searchColors.matchColor`, sets the color for highlighting search results that are not in focus. The second, `vvConfig.searchColors.selectedMatchColor`, sets the color for highlighting the in-focus search result.

Both configurations may be set to a string that contains an rgba color, in the format `"rgba(255,78,0,0.2)"`. This is the default color for search-result highlights. The first three numbers are RGB values to establish the color, and the fourth number is an alpha value--treated like a percent--to define the transparency of the highlight.

New Callbacks

New callbacks have been provided to allow custom code to interact with `VirtualViewer`. In order to set a callback, call `virtualViewer.setCallback("callbackName", callbackFunction)`. This function returns `true` if the callback was set correctly, and `false` if it was not. The callback function should be defined, and should take a single argument object as a parameter. Then, for instance, if the function is declared as `function foo(args) { ... }`, the arguments are accessible in the callback function as `args.firstArgument`.

`VirtualViewer` is responsible for calling the provided callback function appropriately. For instance, `VirtualViewer` will attempt to call the function set to the `"switchToTab"` callback whenever a user switches their tab. Most callbacks do not pay attention to return values, but two new callbacks require a boolean return.

- `annotationChanged` is called whenever the user modifies an annotation; this will fire whenever `VirtualViewer` itself judges that an annotation has been changed and the asterisk appears in the tab name. The following parameters will be provided to the callback in the argument object:
 - `documentId {String}` The ID of the current document whose annotations have been modified
 - `annotationLayerId {String}` The ID of the layer that holds the modified annotation

VirtualViewer 4.13 (and older) New Features and Corrected Issues

- `annotationId {String}` The ID of the modified annotation
- `disableTextContextMenu` is called when the text context menu is about to appear, and if the callback function returns `true`, the context menu will be disabled. This context menu can contain options to copy and cut text if any is selected on the document, to perform OCR, or to close document compare. The callback function will be provided one parameter in the arguments object, and must return a value:
 - `documentId {String}` The ID of the current document that the user is clicking on
 - Return `true` to disable the context menu, and return `false` to allow the context menu to show as normal
- `disablePageManipulationContextMenuOptions` is called when the page thumbnail context menu is shown. If the callback function returns `true`, page manipulation options will be removed from the context menu. Page manipulation options include cut, copy, and delete options; page insertion options; and page selection options. This function is equivalent to setting the configuration option `vvConfig.pageManipulations`, but allows document-by-document granular control. The callback function will be provided one parameter in the arguments object, and must return a value:
 - `documentId {String}` The ID of the current document whose pages the user is clicking on
 - Return `true` to remove the page manipulation options, and return `false` to allow `VirtualViewer` to show or hide the options as normal

Configuration to Disable User Preferences

Now, an administrator can completely disable User Preferences through a new option in `vvConfig`, `vvConfig.disableUserPreferences`. This configuration item can be set to `true` or `false`. If `true`, the User Preferences dialog will be unavailable to users. All configuration items that could be overridden in User Preferences will be drawn from `vvConfig`; users will not be able to override `vvConfig` settings. If not set or set to `false`, User Preferences will behave as normal.

Dynamic Debug Logging

To assist in debugging issues logging can now be toggled into a debug mode without having to change configuration files. Turning on dynamic logging can be done with the client-side call `virtualViewer.loggingOverride(true)` - while this flag is set all requests during that session will log all messages as high priority. This allows finely detailed logs to be created for a specific use case without changing global log configurations.

VirtualViewer 4.13 (and older) New Features and Corrected Issues

Simple Logging Facade (Java)

VirtualViewer Java now implements SLF4J (Simple Logging Facade for Java), a logging abstraction that allows clients to plug in the logging system of their choice. Documentation and examples can be found at <https://www.slf4j.org/docs.html>.

The default logger is still the `java.util.logging` framework. The `init-param logLevel` will only function for the default `java.util.logging` framework - if another logging framework is plugged in using SLF4J, that logging framework's configuration should be used instead.

Common Logging Facade (NET)

VirtualViewer NET now implements `Common.Logging.NET`, a logging abstraction that allows clients to plug in the logging system of their choice. Documentation and examples can be found at <http://net-commons.github.io/common-logging/>.

The default logging functionality is unchanged and is implemented in `Common.Logging`'s configuration as `SnowboundLoggerFactoryAdapter`. The `web.config` parameters `logLevel` and `logToIIS` are also now implemented as arguments in `Common.Logging`'s `web.config` section, although the original `InitParam` arguments will still work for the default logger. If another logging framework is plugged in using `Common.Logging`, that logging framework's configuration should be used instead - `logLevel` and `logToIIS` will only affect the default logger.

Fixes and Changes

Sticky note updates

The double-arrow button to minimize sticky notes will now scale with zoom. Previously, it was possible for the button to be drawn outside the bounds of the sticky note. Now, the button will no longer be larger than the area of the sticky note, and will disappear when the sticky note is zoomed out far enough.

Previously, on a zoomed-out document, it was possible for the size of a minimized sticky note to be larger than the full sticky note. Now, the minimized sticky note will scale properly as the document zooms.

VirtualViewer 4.13 (and older) New Features and Corrected Issues

enableOcr configuration fixed

The enableOcr configuration works again and now defaults to "true" (which will have no effect if your Snowbound license doesn't support OCR). Setting enableOcr to "false" will disable OCR even if your Snowbound license supports it. enableOcr was disabled in 4.12 and replaced with a simple license feature check for OCR.

PDF signature printing issue

Some PDFs have an issue with signatures disappearing when printing via VirtualViewer. We've modified our PDF.js printing solution to fix this issue. If you are encountering this problem, change config.js's disableDirectPDFPrinting to "true" to use our modified PDF.js instead of your browser's.

Misc. Fixes/Changes:

- Fixed issue with inserting annotations + disappearing layers
- Fixed client stack trace in sendDocument
- Added new config parameter, 'consolidateLayerName' to set the default name of a consolidation layer
- Prevent context menus from drawing off-screen when at the boundaries of the viewer
- Fixed document tab showing changes (with an Asterisk) when none were made
- Fixed bookmarks being lost during page manipulations

New Features 4.12

Video

VirtualViewer can now load and play videos, in formats supported by HTML5-compatible browsers. There is no editing or annotation support at this time - videos can only be viewed and downloaded. Video format support will depend on the capabilities of the web browser.

Supported formats

VirtualViewer uses the browser's HTML5 video player to display video, and can play all types of video supported by a browser's player. Most browsers support MP4, WebM and Ogg Vorbis. This browser compatibility chart has more details: https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats

User preferences

There are several new configuration options for displaying video. These options can be viewed and modified in User Preferences:

- Video Autoplay: If enabled, the video will play as soon as it is opened. If disabled, the user must click play in order to start the video.
 - Mute: If enabled, the video will start muted and the user must click to unmute. If disabled, the video will start at full volume.
 - Video Controls: If enabled, the video will have the controls appear on the bottom of the player to control playback, fullscreen, volume, and the ability to download the video. Available options may change depending on the browser you are using. If disabled, video controls will be hidden from view and only accessible through the right-click menu.
 - Video Stretching: If enabled, the video will stretch beyond its original size to fill the viewer. It will keep its aspect ratio, which means the video will not distort as it stretches. It acts like the Fit to Window zoom option, so it will fit to height or width and may not actually fill the viewport. If disabled, the video will not expand beyond its original size and will center in VirtualViewer's main display area.
-

All of these options have equivalent config.js configuration options as defaults.

VirtualViewer 4.13 (and older) New Features and Corrected Issues

Supported features with video

We currently support the following actions with video:

- Opening and viewing video.
- Downloading the video. This cannot be disabled.
- Change video viewing size (original size, fit to window and fullscreen).

The following are some things that are limitations of the HTML5 Video Player:

- Some browsers may not be able to seek, they just play from start to finish. Firefox had no issues seeking, Chrome did.
- Fast Forward and Rewind aren't baked into HTML5 Video Player.

Add configuration to auto-resize only sticky notes

When the configuration parameter `vvConfig.autoResizeTextAnnotations` is set to true, the viewer automatically resizes text annotations to fit the annotation text. If `vvConfig.autoConfirmTextAnnotations` is also set to true, the text annotation will change its size as the user types.

Now, there is a new configuration parameter `vvConfig.autoResizeStickyNoteAnnotations`. This fine-tunes the configuration control. `vvConfig.autoResizeTextAnnotations` will now only affect text annotations, and `vvConfig.autoResizeStickyNoteAnnotations` will only affect sticky note annotations.

Set document display name API

A new Javascript API `virtualViewer.setDisplayName(newDisplayName, documentID)` will set the given document ID to the new display name. The display name will update on the document tab and document thumbnail.

Load document model asynchronously

Highlight annotation button currently in use

Adjust toolbars intelligently based on how many buttons are visible

Redaction Navigation

Enhanced redaction navigation has been added to VirtualViewer. When the configuration for `vvConfig.showAnnNavToggle` is set to true, the previous navigation buttons are displayed as well as a set of radio buttons. The radio buttons are for the two navigation modes and display the annotation and

VirtualViewer 4.13 (and older) New Features and Corrected Issues

redaction counts for the document. When the annotation navigation mode is selected, navigation occurs as it did in the past. When redaction navigation is selected, it goes from redaction to redaction throughout the whole document. The redactions are selected and are focused on. The filter pages button does not work in redaction navigation mode and is disabled.

Page Manipulation with Bookmarks

VirtualViewer now retains bookmarks created on pages of a document as they are subject to page manipulations. The bookmarks would remain with the page in both reordering pages, cut/paste to other documents and cut/copy to new document.

Cache-seeding Support (Java)

There is a new client side API that allows the server to pre-cache images for future use. By calling `virtualViewer.seedCache(documentId, pages, clientInstanceId)`, the user can get pages ready on the server, allowing for quicker retrieval.

Fixes and changes

VirtualViewer initialization API are easier to work with

The API function `beforeVirtualViewerInit` is called before the VirtualViewer object is initialized, and the API function `afterVirtualViewerInit` is called after; these functions are intended to be defined by customers to easily run code as VirtualViewer starts up. Previously customers would have to define these API functions after `index.html` was loaded; now, they can be defined any time.

Preserve document scroll when zooming

Previously, zooming in and out would cause the document to jump to the top of the current page. Now, older functionality is restored, and the document will stay in the same scroll position while zooming in and out.

Preserve document scroll and zoom when switching between tabs

When switching between open document tabs in the viewer, documents will now stay at the same zoom and scroll position that the user set, rather than zooming to the default zoom and going to the top of the page. If the configuration parameter `fitLastBetweenDocuments` is set to true, the document will still apply the zoom of the current document to the next document opened.

disableUploadDocument parameter moved to server

The `config.js` parameter to disable the Upload Document functionality has been replaced with a server setting. This will completely disable the service endpoint for upload so that clever users couldn't bypass the UI to "force" a document into the system. In prior releases we encouraged this filtering occur in the Content Handler as needed. But if you want to disable it altogether, you now have an easy method to do so.

Removed sample content handlers from VirtualViewer distributable

Some sample content handlers were removed from compiled VirtualViewer code. They will no longer be accessible to use - any in use should be replaced by a customized version of the default content handler.

InputStreamContentHandler

FileAndURLRetriever

MergedImageContentHandler

MultiContentElementContentHandler

Public print API

The parameters for the printDocument() method have changed. The new parameters are as follow,

- {String} The documentId to print, must be of a document open in the viewer
 - {Boolean} If true, a PDF will be exported to a file. The user will be present with a save dialog
 - {Boolean} If true, the annotations will be printed.
 - {Boolean} Whether or not to burn in redactions.
 - {Boolean} Whether or not to include redaction tags (only used when includeRedactions is true).
 - {Boolean} Whether or not to include watermarks.
 - {Boolean} Whether or not to include document notes.
 - {String} Either "all", "complex" or "current".
 - {String} A range of pages numbers to export (only used for "complex" pageRangeType).
-

Misc. Fixes/Changes:

- Improved the responsiveness of the toolbars
- Added a close button to the split screen/document compare panel
- Fixed an issue with the autoLayerPrefix when there were existing autolayers
- Previously, if you created an annotation layer in the Layer Manager dialog and clicked the dialog boxes OK box (instead of the new Layer) the layer would not be created.
- Fixed issue with annotation layers and page manipulations
- Fixed several minor/cosmetic issues with search UI
- Load the document model asynchronously, to improve responsiveness
- Make sure the document model has been loaded before requesting the image from the server
- Fixed tab naming after closing document compare panel
- Fixed issue with watermarks and reordering pages
- Confirm user wants to save when closing the browser window/tab
- Fixed an issue with image buffering

VirtualViewer 4.13 (and older) New Features and Corrected Issues

- Fixed issue with page-change callbacks where they were firing even if the page didn't actually change (like calling `firstPage()` when you were already on page one).
-

New Features 4.11

Document Compare

Summary

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.

Entering the workflow

To start comparing documents, first open a document. This document will be the "original" or "old" document in the comparison. Navigate to the document tab in the right-hand thumbnail pane, and right-click the document thumbnail. Select the option "Document Comparison" in the menu that appears.

This option will open your second document in a new pane of the viewer, splitting the view in half. It will also start the process of document comparison. In the right sidebar, a list of pages will appear. Clicking on one of those pages will reveal a list of differences between the documents.

Types of changes

Consider a document that has the first line "To be or not to be," open in the left-hand pane as the original document. Choosing the second document, the user opens a document with the first line "To think or not to think." Document comparison interprets that "be" has been removed and "think" has been added. If this were reversed, and the "to think or not to think" document were open on the left, document compare would declare that "think" has been removed and "be" has been added.

Navigating document compare

Changes in the sidebar are displayed page by page. Click on the "Page 1" text to see all the changes on page 1; click again to collapse those changes away. Clicking on the text of a change in the sidebar will scroll the document to where the text is actually located on the page.

If a page has no changes, it will display "There are no changes to display." If page text has not loaded yet, a loading gif will display. There will also be a "Load More" button at the bottom of the sidebar. If

VirtualViewer 4.13 (and older) New Features and Corrected Issues

pages have not been compared, it means their text has not been loaded; either clicking the Load More button or scrolling through the documents will load more page text and thus more document comparison. Page loading and document compare is incremental to preserve performance on long documents.

At the top of the document compare tab in the sidebar, there are three small buttons for controlling document compare. The first, with an icon like an eye, will toggle whether the red and green highlighting on the document is visible. The second, with an icon like a padlock, will lock document scrolling. Scrolling up or down in one document will scroll the other visible document just as much. Finally, the refresh button will re-run and re-display document compare.

Exiting the workflow

Right-clicking on the right-hand document will bring up a menu with the option "Close Document Comparison." This will close the second pane, returning to the full-screen view. The document comparison tab in the right-hand thumbnail pane will also be hidden.

Using OCR with document comparison

If OCR is set up, it is possible to compare documents using OCR. This will occur automatically. If document comparison is initiated and at least one of the documents contains no text, the user may request that OCR is performed. The resulting text will be compared. If OCR is not configured, it will not be attempted, and documents without text simply cannot be compared.

API

```
virtualViewer.compareDocuments(firstDocumentID, secondDocumentID, { splitScreenDirection } )
```

Parameters:

- firstDocumentID {String} This document will be opened in the left pane, as the "original" document.
- secondDocumentID {String} This document will be opened in the right pane, as the "new" or "revised" document.
- options {Object} This options object does not need to be passed in, and currently contains one optional parameter; it may be modified later to contain more. To pass data in through the options object, create an object with a key-value pair:

```
virtualViewer.compareDocuments("myOldDocument.pdf", "myNewDocument.pdf", {  
splitScreenDirection: "vertical" } );
```

VirtualViewer 4.13 (and older) New Features and Corrected Issues

- `splitScreenDirection` {String, "horizontal" or "vertical"} The default value is "horizontal". The value "horizontal" will display document panes side-by-side. Pass in the value "vertical" to display the document panes one on top of the other.

Returns: undefined

The API `compareDocuments` will open both of the provided document IDs in the viewer. The first document ID will be the "original," left-hand document, while the second will be the right-hand document. The final parameter is an options object, which can pass in optional parameters. There is one optional parameter, `splitScreenDirection`.

`virtualViewer.getDocumentCompareReport()`

Parameters: `onReportCompleteCallback`

Returns: A string containing the document comparison data, if no callback is provided as a parameter.

The API `getDocumentCompareReport` will provide an HTML report of all the changes between the two open documents in a unified diff. The report will be provided as an argument to the `onReportCompleteCallback` given as a parameter, or returned directly from the function `getDocumentCompareReport` if no callback is provided. Providing a callback is recommended, since then document compare will be able to finish loading and processing asynchronously.

The return string is a `<div>` element; each changed item is a `` within that `<div>`. This includes unchanged text.

- Spans containing unchanged text will have the class name `documentCompareReportNoChange`
- Spans containing text removed from the original will have the class name `documentCompareReportTextRemoved`
- Spans containing added text will have the class name `documentCompareReportTextAdded`

This report does not contain styling.

Callback Event Manager

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

API

VirtualViewer 4.13 (and older) New Features and Corrected Issues

```
virtualViewer.setCallback('callbackString', callbackFunction{})
```

Parameters:

- *callbackString*: The following are the callback names. You pass in the string and it will properly set that callback.
- *callbackFunction* Pass in the function that you want to be called. It should accept one Arguments object that has the values listed next to the callback below.
- *onDocumentLoad* documentId
- *saveDocument* documentId, clientInstanceId, documentIdToReload
- *saveAnnotation* documentId, clientInstanceId, documentIdToReload
- *saveAsDocument* oldDocumentId, newDocumentId, clientInstanceId
- *uploadDocument* uploadedDocumentId, clientInstanceId
- *sendDocument*
- *pageChange* page
- *pageCopied* pages
- *pagePasted* pages
- *pageDeleted* pages
- *rotation* page, degrees
- *textSelect* text
- *imageLoadFinished*
- *imageLoadRequested*
- *annotationCreationCallback* type

Sticky Note Background Color Support

You can now set the background color of your sticky notes. You can set the default preference in the config.js, a personal user preference in the UserPreferences menu, and individual background colors for stickies in their annotation properties menu.

User Preference Option for Default Thumbnail Tab

Now in the user preference General Preferences tab there is a section that lets you select which tab of the thumbnail panel is the default upon opening the viewer. This should help users get to the navigation they need faster.

Copy Annotations Across Documents

VirtualViewer 4.13 (and older) New Features and Corrected Issues

We expanded the copy/paste annotation functionality to work across documents in the viewer. This is only implemented for the current session, you can't copy and paste an annotation between different windows of the viewer. This is strictly for switching between tabs in the viewer.

Get Page Dimensions

You can make an API call to return the actual dimensions of an image.

API

```
virtualViewer.getOriginalPageDimensions()
```

Parameters: None

Returns: An object that contains the height and width of the image. E.g. *{height: 595, width: 679}*

Require.js

We now use Require.js to compile and load our javascript code. Each of our javascript files is now an AMD module, and VirtualViewer javascript code will be delivered to the browser in concatenated files to reduce latency from loading multiple script files. The code delivered to the browser will look very different from previous versions of VirtualViewer.

API

To make it easier to hook into VirtualViewer as it initializes, we look for two functions as VirtualViewer starts up. Define these functions in the global space of the iframe or window that VirtualViewer is running in.

```
beforeVirtualViewerInit
```

This function is called before VirtualViewer calls any initialization function. The virtualViewer object containing API will exist, but API functions may perform unexpectedly before initialization functions. beforeVirtualViewerInit is an appropriate place to call an alternate VirtualViewer initialization API, or initialize other objects.

Parameters: None

VirtualViewer 4.13 (and older) New Features and Corrected Issues

Returns: If you return true from this function, VirtualViewer will continue with normal initialization procedures. Return false if you are making calls to initialize VirtualViewer in beforeVirtualViewerInit, as these should override VirtualViewer's initialization.

afterVirtualViewerInit

This function is called after VirtualViewer has completed initialization. This is an appropriate place to assign callbacks, and perform initialization tasks that depend on VirtualViewer API.

Parameters: None

Returns: None

Thumbnail name customization

Thumbnail names can now be specified within the Pages tab

Code Sample:

```
virtualViewer.getCurrentState().setPageDisplayNames(["My First Page", "My Second Page"]);
```

Changing the Document display name

For changing the display name for the open Document tab, the individual pages within the Pages tab, & the hover tooltip over individual pages..

```
virtualViewer.getCurrentState().setDisplayname("newDisplayName")
```

Code Sample:

```
virtualViewer.setCallback("onDocumentLoad", function(event) {  
  var state = virtualViewer.getStateForDocumentId(event.documentId);  
  state.setDisplayName("My Document: " + event.documentId);  
});
```

VirtualViewer 4.13 (and older) New Features and Corrected Issues

API call to toggle annotations on/off

virtualViewer.toggleAnnotationVisibility(show)

'show' is an optional boolean: if not provided, the function will toggle back and forth; if set to true/false it will make the annotations visible/invisible.

VirtualViewer 4.11 Bug Fixes

Annotation Revision History saving (fixed)

Emailing as original sends a tif (fixed)

Two Clicks to Move Annotation (fixed)

Mapping Alfresco permissions to Snowbound - document available from customer support

Annotation paste / save no longer fails but may be misplaced - still in development

ContentHandlerInput input now includes a `HttpServletRequest` object in its table

pathnames missing within a war file corrected -

XHTML: File hangs - XHTML Files over 1000 pages long may cause problems for the viewer

VirtualViewer 4.13 (and older) New Features and Corrected Issues

New Features of VirtualViewer 4.10

Mobile Device Control Improvements

The user can pan on an image, if it's zoomed in, in all directions (imageScrollbars should be set to false in config.js for this to work).

Using two fingers, the user can pinch to zoom in or zoom out on the document. The zoom motion is not animated so the increase/decrease in size will happen when the user is done pinching.

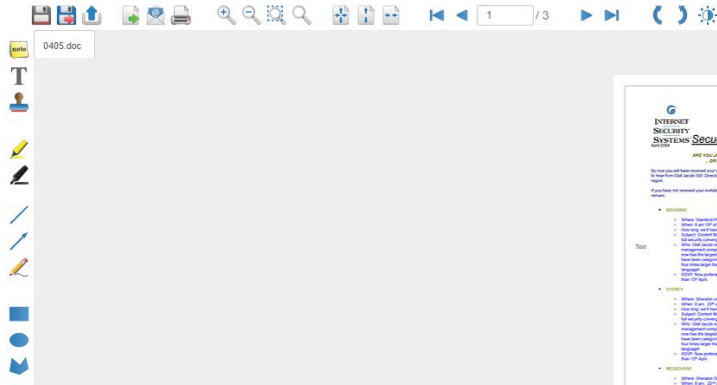


The viewer now works more elegantly in small iframes, on low-resolution monitors, and when browser windows resized smaller.

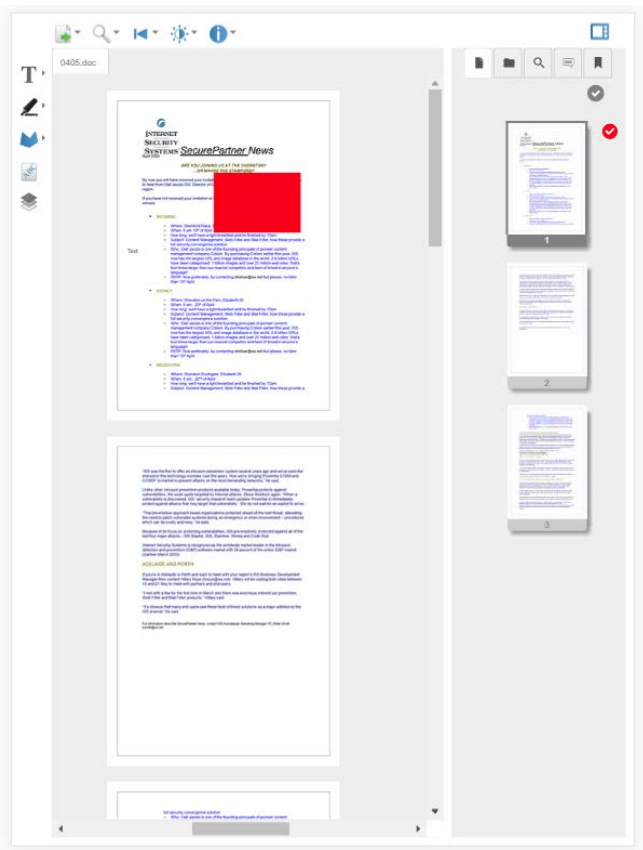
Toolbar details

On large screens, the toolbars look exactly the same as before:

VirtualViewer 4.13 (and older) New Features and Corrected Issues



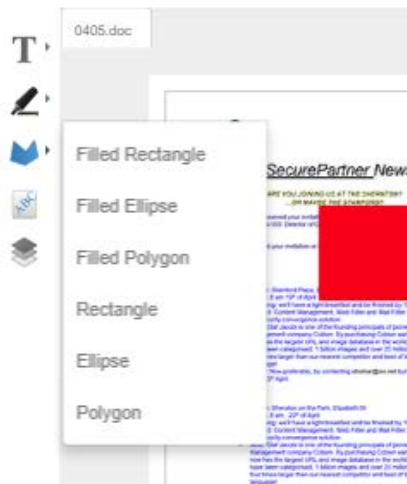
But as you shrink down, buttons collapse into menus. This happens as you shrink your monitor size, lower your resolution, or even shrink your browser. It's especially important for embedding the viewer in small iframes--like Alfresco--or for using the viewer on an iPad:



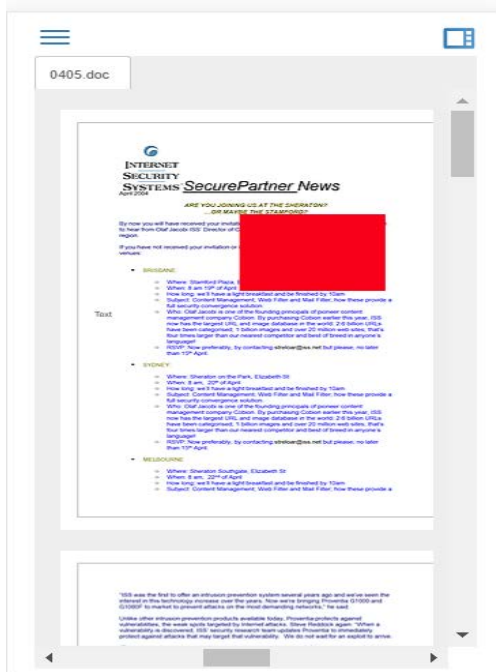
(Note that these icons are temporary.)

Here, you can see what the dropdown menus look like:

VirtualViewer 4.13 (and older) New Features and Corrected Issues

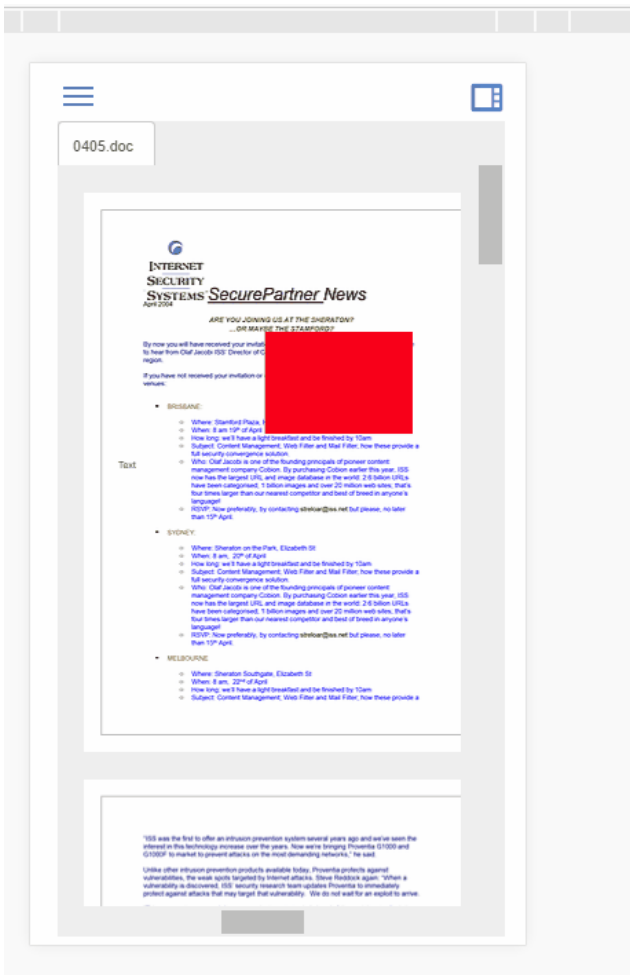
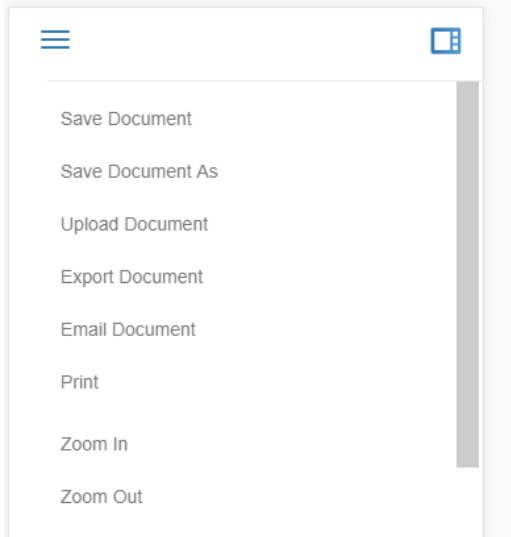


As you shrink down even further, into phone real estate or into the smallest your browser can get, we have even less space on the toolbar. So, we stuff all the options into a large menu, reachable by the three lines in the top left (called a "hamburger button"):



VirtualViewer 4.13 (and older) New Features and Corrected Issues

This menu takes up the entire screen when open:



VirtualViewer 4.13 (and older) New Features and Corrected Issues

Toolbar buttons are also now highly configurable.

A deep dive into the configuration

The file `user-config/toolbar-config.js` now defines everything about the toolbars. Customer admins can go in and manage this instead of modifying `index.html` directly: VirtualViewer code goes into this configuration file and creates a toolbar button for every configuration entry that it sees.

There are three parts to `toolbar-config.js`. First, we have the two big lists: `imageToolbarButtons` for the top toolbar and `annotationToolbarButtons` for the left-hand toolbar. Their names should categorically not be modified, particularly not in the final return statement in `toolbar-config.js`.

Take, for instance, `imageToolbarButtons`. It looks like this:

```
var imageToolbarButtons = {  
  
  "buttonKey": {button configuration},  
  
  "anotherButtonKey": {button configuration}  
  
};
```

Each "buttonKey" is a unique, descriptive key for the toolbar button. Unique because it's defining the button in the configuration, and also because it's used in the HTML. It should be letters only, and it is good practice for customers to put a prefix on this key--for instance Big Company, Inc may call a key "bcMyKey."

This is the same structure as the annotation toolbar list of buttons. Each button key should also be unique between the two lists--don't put "vvMyNewButton" as a key in the image toolbar configuration and in the annotation toolbar configuration.

Now we can look at the button configuration. In the curly brackets, there are a few items:

-localizeKey: This is a string referring to a locale file, where the text that appears in the title--what appears in the tooltip and in the dropdowns--is stored. If a customer is creating a whole new button that doesn't have an entry in a locale file, this field can be left blank, but the "name" field *must* be filled in or the button will appear blank in dropdowns.

-name: This is a string with the name of a toolbar button. VirtualViewer buttons do not use this property, since they use localized strings, but customers without an entry in a locale file must put the name of their toolbar button in this field. This should be something short and descriptive.

VirtualViewer 4.13 (and older) New Features and Corrected Issues

-clickHandler: This is the function that will be called when a user clicks on the toolbar button.

-iconImage: This is a string representing a URL to an icon image. VirtualViewer buttons use CSS files to assign icon images, but customers can list an image here and it will be used.

-addSeparatorAfter: This is a boolean value. If a group of buttons is too long, it may be necessary to add a bit of whitespace after a button to visually break up the group when the toolbar is fully expanded. If this is true, the viewer will pop some white space after the button. It is not necessary to specify "false" on this item.

-groupId: This is a string, and it refers to a group key. Groups are also configured in toolbar-config.js. If a button is in group A, it will be placed in group A in the order it appears--if the Sticky Note button is before the Rubber Stamp button in the annotation toolbar list, and they both belong to group A, the Sticky Note button will appear before the Rubber Stamp button in the UI. If an item is in the annotation toolbar list, it should be assigned to an annotation toolbar group. Additionally, this is optional--if a button has no groupId, it will essentially be in an "uncategorized" list. It will appear at the end of the toolbar and will not be included in a dropdown. (See the Layer Manager and Watermarks buttons as default examples.)

The final part of toolbar-config.js is the group configuration list, called toolbarButtonLogicalGroups. This list looks very similar to the button configuration lists. What it will do is define all the buckets that the toolbar buttons can go into.

Again, each group has a unique string key followed by a configuration object:

```
var toolbarButtonLogicalGroups = {  
  
  "myGroupKey": { configuration },  
  
  "myOtherGroupKey": { configuration }  
  
};
```

The configuration items are as follows:

-localizeKey: This is a string referring to a locale file, where the text that appears in the tooltip of the dropdown button is stored. This can be left blank.

-groupTitle: Like "name" for the buttons, this is a string with the name of the group in it. For instance, "Preferences" or "Edit" or "File." The localize key will handle this, but a customer without a locale entry could use "groupTitle".

VirtualViewer 4.13 (and older) New Features and Corrected Issues

-iconImage: This is a string representing a URL to an icon image. VirtualViewer buttons use CSS files to assign icon images, but customers can list an image here and it will be used.

-annotationToolbar: This is a boolean flag that chooses where to put the group. If true, the group is an annotation toolbar group; if false or absent, the group is in the top toolbar. Annotation toolbar buttons should only be placed in annotation groups, and similarly, image toolbar buttons should only be placed in image toolbar groups.

New strings in vv-en.json

utilityToolbar.fileGroup: File

utilityToolbar.zoomGroup: Zoom

utilityToolbar.pagesGroup: Pages

utilityToolbar.pageManipulationGroup: Page Manipulation

utilityToolbar.infoGroup: Info and Settings

annToolbar.textAndStampsGroup: Text and Stamps

annToolbar.markupGroup: Markup

annToolbar.shapesGroup: Shapes

Unsupported Features on Mobile Devices

Drag and Drop

Search and redact are disabled if redaction is disabled

Document comparison, when offered, will not be supported for mobile devices

Though several mobile devices have been approved, we cannot guarantee that all mobile devices are supported.

Alfresco Quickshare Support

A logged-in user can create or close a public quickshare link through Alfresco. If a logged-in user accesses that quickshare link, they can see and modify the document through VirtualViewer as normal; if a guest or unauthenticated user accesses the quickshare link, they will be able to view the document but not save any modifications to the document (or annotations, notes, etc).

VirtualViewer 4.13 (and older) New Features and Corrected Issues

Currently the unauthenticated user gets the full VirtualViewer UI with a readable error message if they take any action that tries to save the document. In the future they should get a limited UI that doesn't present those options.

Alfresco Watermark Support

Added support for Watermarks in the Alfresco version of VirtualViewer allowing saving watermarks back into the Alfresco repository. All supported features or functions remain.

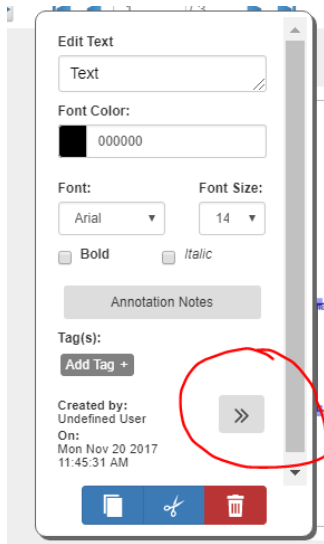
Revision history for Annotation Create Date/Time

Use Case:

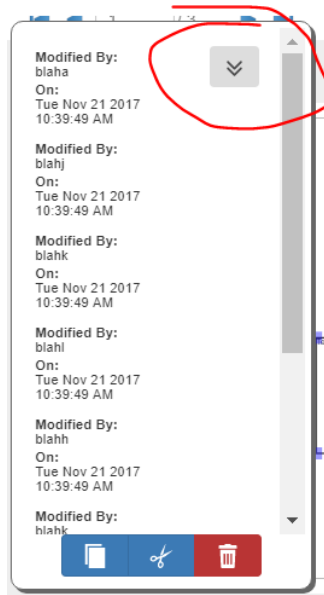
1. User 1 creates an sticky note on page 1 of a document, the userId/date/time are recorded
 2. User 2 edits that same sticky note (color, size, placement, any change really....ect)
 3. The userId/date/time are updated in a scrolling list reflecting each edit to that object.
-
- Works for all annotations,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



Filenet F_CREATOR Tag Support Added

NET 4.5.2+ Requirement Added for TLS 1.2 support

OCR Integration (beta)

VirtualViewer 4.13 (and older) New Features and Corrected Issues

This is a beta version of the OCR option in VirtualViewer. The final version is expected in the VirtualViewer v4.11 release where some alignment issues and ability to work with additional input file formats will be added. We also expect to offer a choice of OCR recognition engines in 2018.

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 OCR'd. 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 OCR'd 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'd 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 raster. Saving to a PDF file is an option. Additional language support can be added by the customer.

The two new parameters necessary to enable OCR are in web.xml (web.config for .NET):

- `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"`.

Fixed Customer Issues in VV 4.10

Small screen issues fixed

- The document no longer overlaps the thumbnail panel when the viewing canvas gets very small
- Thumbnails work on iOS devices, annotations may be drawn and modified on all sizes of screen
- The thumbnail pane starts out hidden if it's going to take up too much screen real estate
- Modal dialogs are usable on small screens
- Annotation context menu is usable on small screens
- Scrolling works on small screens

Other Fixed Customer Issues in VV 4.10

- IOS Mobile: Export to Tiff/Original opens only the 1st page in new tab
 - This is an issue with IOS where it only displays the first page of a multi-page TIFF. The other pages are preserved and viewable on other applications. (This is also true when viewing such documents in Gmail.
- VV Net 4.9: Constant reloading of thumbnails
- FileAndURLRetriever class missing from VV Java v4.9
- VV: PDF Arial to Times New Roman conversion error fixed
- Tabs, canvas, and thumbs can lose sync
- Annotation Highlight Rectangle does not use highlightFillColor
- Annotations are not burnt in via export
- Text Ann Color Selection
- Confirm Changes dialog box not appearing
- Snowbound annotation tags are overwritten
- Page size increased after redaction applied
- Filenet annotation values are overwritten
- PDF document comments are not being displayed
- VV Net HTML plug issue
- PaperPort 12 PDF: Large file stalls or fails to load
- VV .NET cannot get files over HTTPS TLS 1.2 on framework 4.5
- Color is not displayed and size is shrunk in pdf document
- Cannot view annotation with permission PERM_VIEW = 40

END OF RELEASE NOTES

VirtualViewer 4.13 (and older) New Features and Corrected Issues